tanulmányok

203/1988

MTA Számitástechnikai és Automatizálási Kutató Intézet Budapest

MAGYAR TUDOMÁNYOS AKADÉMIA SZÁMÍTÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI
KUTATÓ INTÉZETE
COMPUTER AND AUTOMATION INSTITUTE, HUNGARIAN ACADEMY OF SCIENCES
Исследовательский Институт Вычислительной Техники и Автоматизации
Венгерской Академии Наук

РГ - 25, КНВВТ
ПРОБЛЕМы И ИНСТРУМЕНТАРИИ ИНТЕГРАЦИИ ИНФОРМАЦИОННыХ СИСТЕМ
СБОРНИК

1987

KNVVT WG-25
Problems and tools of the integration of information systems
PROCEEDINGS

1987

Tanulmányok 203/1988
Studies 203/1988

H-1502 Budapest PO Box 63. XI. Kende u. 13-17. Hungary

A kiadásért felelõs:

KEVICZKY LÁSZLÓ

ISBN 963 311 244 3
ISSN 0324-2951

СОДЕРЖАНАНИЕ - CONTENTS

П Р Е Д И С Л О В И Е

Проблемная комиссия многостороннего сотрудничества Академий наук социалистических стран "Научные вопросы вычислительной техники" (КНВВТ) создана согласно решению Первого совещания Академий наук социалистических стран /Варшава, май 1962-го года/.

В рамках КНВВТ проводятся фундаментальные и прикладные исследования в следующих направлениях:

- Вычислительные методы,
- Методы пргоаммирования,
- Теоретические основы вычислительной техники,
- Применение математических методов и ЭВМ .

Одной из главных форм сотрудничества являются рабочие группы /РГ/, которые создаются для разработки особо актуальных и перспективных задач в рамках тематики КНВВТ целью решения конкретных задач научно-технического характера, составления обзоров и анализов состоянии определенных научных направлений, разработки технических проектов для создания конкретных систем или пакетов программ.

Создание рабочей группы РГ-25 "Проблемы и инструментарии интеграции информационных систем" было утвержде-

но согласно предложению, изготовленного представителями из НРБ, ВНР, ГДР, Кубы, ПНР, СРР, СССР и ЧССР на XXII заседании КНВВТ /Будапешт, май 1986-го года/.

Ответственность для РГ-25 поручена Болгарской АН. Как председателем предложена и утверждена Румяна Киркова, а как зам. председателем - Геннадий К. Столяров /АН СССР/. КНВВТ предложила также базой РГ-25 стать уже сработанный коллектив ученных и специалистов рабочей группы РГ-19 "Системы управления распределенными базами данных", которая успешно завершила свою работу в 1986-ом году.

В 1987-ом году РГ-25 провела два рабочих совещаний: в г.София (Болгария) с 6 по 11 апреля и в г.Ефорие-норд (Румыния) с 14 по 19 сентября. В эти совещания принимали участие представители из НРБ, ВНР, ГДР, ПНР, СРР, СССР и ЧССР всего 37 участников.

В результате проведенных обсуждений сейчась в рамках РГ-25 развиваются следующие научные направления:

1. Методы исследования, создания и сопровождения информационных систем
   /ответственные АН: ПНР и ЧССР/
2. Информационные системы для мини и микро-ЭВМ
   / ответственные АН: НРБ и СРР/
3. Информационные системы в локальных сетях
   / ответственные АН: НРБ и СССР/
4. Методы и инструментарии интеграции информационных систем
   /ответственные АН: ГДР и СССР/

По этим направлениям работают утвержденные РГ-25 национальные и интернациональные коллективы. Они предоставляют на утверждение планы своих исследований и отчитывают свои научные результаты в РГ-25. Для этой цели параллельно каждого рабочего совещания проводится и научная сессия по тематике рабочей группы.

Участники РГ-25 считают очень важно оформление результатов исследований имея ввиду, что этими результатами широко будут пользоваться специалисты в в сотрудничающих стран. С этой целью было принято решение: периодически подгатавливать и выдавать выпуски Сборника научно-исследовательских и прикладных работ, завершенных в рамках или со содействием РГ-25. Ответственность на опубликование отдельных выпусков этого сборника до сих пор приняли на себя ВНР и СРР.

Перед Вами первый выпуск Сборника научно-исследовательских и прикладных работ по тематике РГ-25,КНВВТ. В нем включены все материалы, которые обсуждались во время двух научных сессии и были получены к середине ноября с.г. в адрес председателя рабочей группы.

Участники РГ-25 выражают свою благодарность Руководству исследовательского института вычислительной техники и автоматизации /ИИВТА/ Венгерской АН за предоставленную возможность опубликовать первый выпуск сборника РГ-25, а также редакционной группе, которая тщательно подготовила и отпечатала все полученные материалы.

ноябрь 1987-го года
София - Будапешт

Румяна Киркова
Председатель РГ-25

MTA SZTAKI Tanulmányok 203/1988 p. 9-21

ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ БАЗЫ ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ С ПЕРЕМЕННОЙ СТРУКТУРОЙ

Галина Виткова

Центральный вычислительный институт
ЧСАН

1. ОБЩИЕ ЗАМЕЧАНИЯ

Существует широкий класс задач, где использование банков данных представляется целесообразным и даже необходимым, который можно характеризовать следующим образом.

В начале решения поставленной задачи известна лишь небольшая часть данных (так называемые исходные данные), которые предполагается хранить в базе данных.

Постепенно по мере решения задачи накапливаются так называемые промежуточные данные, т.е. результаты расчетов, которые также целесообразно по той или иной причине поместить в базу данных. Причем в ходе решения задачи становится ясным, что часть промежуточных данных имеет смысл удалить из базы данных. Объем промежуточных данных может во много раз превышать объем исходных данных.

И, наконец, как результат окончательного решения задачи возникают конечные или результирующие данные. Отсюда видно, что изменчивость структуры данных является существенной чертой рассматриваемого класса задач.

Типичные примером таких данных являются экспериментальные данные и результаты их обработки на ЭВМ.

Проанализировав возможности использования систем баз данных при решении задач описанного класса, мы пришли к следующему заключению [10, 11]:

1. Для нашей цели наиболее целесообразно использование

СУБД общего назначения прежде всего потому, что эти системы обеспечивают оптимальное управление данными, а развитие и совершенствование самих СУБД обеспечивается организациями, поставляющими их.

2. Реляционная модель данных наиболее пригодна в нашем случае, т.к.:
   - достаточна проста и понятна конечным пользователям (т.е. исследователям);
   - имеет средства для удобного доступа к данным и для их реорганизации (непроцедуральный язык запросов);
   - позволяет создавать средства, сравнительно эффективные, для автоматизированного проектирования логической структуры базы данных.

Остановимся очень коротко на основных понятиях из теории проектирования реляционных баз данных, которые были непосредственно использованы для создания так называемого комплексного алгоритма для проектирования базы данных с переменной структурой.

2. КРАТКИЙ ОБЗОР ОСНОВНЫХ ТЕОРЕТИЧЕСКИХ ПОНЯТИЙ

Вместе с большинством авторов [2, 3, 4, 5, 9] в качестве исходной предпосылки принимаем принцип существования универсального отношения.

Не вдаваясь в подробности, можно сказать, что при таком подходе, все отношения, хранящиеся в рассматриваемой базе данных, являются проекцией уже упомянутого универсального отношения (иное и очень интересное толкование этого принципа дается например в [6]). В этих условиях задача проектирования логической структуры реляционной базы данных формулируется следующим образом.

Пусть задана схема универсального отношения

$$S = (U, G),$$

где

U - множество атрибутов универсального отношения;

G - множество ограничений целостности.

Нужно найти схему реляционной базы данных $R=\{R_i(U_i,F_i)\mid i=1,\ldots,n\}$, которая эквивалентна с S с точки зрения представления информации, но обладает "лучшими" свойствами или качеством, чем S .

Большинство авторов различает следующие критерии эквивалентного представления информации [2, 5, 8]:

-R содержит ту же информацию как и S , если R определена на тех же атрибутах как S , и в R сохранены те же зависимости между данными как и в S , т.е. R обладает свойством покрытия S .

- R эквивалентна с S по представляемой информации, если R определена на тех же атрибутах как и S , а база данных со схемой R содержит те же данные, как и экземпляр универального отношения S , т.е. R обладает свойством соединения без потерь.

-R представляет ту же информацию что и S, если R определена на тех же атрибутах что и S , в R сохранены все зависимости между данными, которые определены в S и, наконец, экземпляр базы данных со схемой R содержит те же данные, что и экземпляр универсального отношения, т.е. R обладает свойством сохранения зависимостей из S и свойством соединения без потерь.

Качество отношение можно оценивать степенью его нормализации: чем выше нормальная форма, в которой находится отношение, тем оно "качественнее" в первую очередь с точки зрения устранения так называемых аномалий при его актуализации [5, 7].

Для нашего класса задач в реляционной схеме R необходимо сохранить все зависимости и одновременно обеспечить соединение без потерь, т.е. необходимо выполнить последний критерий.

Что касается степени нормализации, то мы остановились на так называемой улучшенной третьей нормальной форме (далее УТНФ) по следующим соображениям: во-первых, в УТНФ можно перевести любое отношение, а во-вторых, среди экспериментальных данных мы пока обнаружили (исходя из имеющегося опыта обработки этих данных с использованием СУБД) лишь функциональные зависимости.

УТНФ была предложена в [7] ввиду того, что определение ТНФ, данное Коддом, относится лишь к одному отношению, а каждая база данных состоит из нескольких отношений. Если не учесть это обстоятельство, то при актуализации базы данных, в которой каждое отношение по отдельности находится в ТНФ, могут случиться аномалии, как это видно из примера в [7]:

Пример

Пусть задано множество атрибутов $U = \{A, B, C, D, E, F\}$ и множество функциональных зависимостей $G = \{ AB \rightarrow CD,\ A \rightarrow E,\ B \rightarrow F,\ EF \rightarrow C \}$. Предположим, что на основе этих исходных данных была спроектирована реляционная схема $R = \{ R_1, R_2, R_3, R_4 \}$, в которой отношению R_1 принадлежит множество атрибутов $U_1 = \{ A, B, C, D \}$ и множество ключей $K_1 = \{AB\}$, R_2 с $U_2 = \{A, E\}$ и $K_2 = \{A\}$, R_3 с $U_3 = \{B, F\}$ и $K_3 = \{B\}$ и наконец R_4 с $U_4 = \{B, F, C\}$ и $K_4 = \{EF\}$. Схема R обладает свойством сохранения функциональных зависимостей и свойством соединения без потерь, и каждая схема $R_i \in R$ находится в ТНФ. Предположим, что имеется экземпляр R с кортежами $R_1 (a_1, b_1, c_1, d_1)$, $R_2 (a_1, e_1)$, $R_3 (b_1, f_1)$ и $R_4 (e_1, f_1, c_2)$. Тогда значение группы атрибутов AB, т.е. (a_1, b_1), можно получить по значению атрибута C (т.е. c_2) с использованием зависимостей, действительных вне отношения R_1. Если $c_1 \neq c_2$, тогда будет нарушена консистенция БД с реляционной схемой R.

В теории проектирования реляционных баз данных до сих пор различают два подхода: синтез и декомпозиция, хотя в последнее время разница между ними заметно стирается. Достоинством алгоритмов синтеза является однозначность решения,

сохранение зависимостей в любом случае, соединимость без потерь, обеспечиваемая в случае необходимости. Недостатком этих алгоритмов является тот факт, что они не обобщены для многозначных зависимостей. Алгоритмы декомпозиции годятся как для функциональных, так и для многозначных зависимостей. В этом их большое достоинство. Но они не приводят к однозначному решению, не обеспечивают сохранение зависимостей и их вычислительная сложность высока (см. напр. [8]). С учетом свойств алгоритмов синтеза и декомпозиции и имея в виду характеристику нашего класса задач, мы сделали выбор в пользу алгоритмов синтеза [12].

Резюмируя сказанное, в предлагаемой ниже методике проектирования реляционной базы данных с переменной структурой мы приняли за основу третий критерий эквивалентности в представлении информации, УТНФ как основной критерий качества реляционной схемы и алгоритмы синтеза как платформу комплексного алгоритма.

3. МЕТОДИЧЕСКАЯ ОСНОВА КОМПЛЕКСНОГО АЛГОРИТМА

В проектировании логической структуры базы данных с переменной структурой различаем две составляющие:

А/ Основную процедуру, задачу которой можно сформулировать следующим образом:

Пусть задано на основе семантики экспериментальных данных множество функциональных зависимостей $F = \{ f_1, \ldots, f_n \}$, определенное на множестве атрибутов U, характеризующих свойства объектов, описываемых этими данными. В соответствии с предпосылкой о существовании универсального отношения обозначим схему этого отношения в виде $S(U,F)$, где U - множество атрибутов этого отношения, а F - вышеупомянутое множество функциональных зависимостей. Требуется найти реляционную схему $R = \{ R_i (U_i, F_i) \mid i=1,\ldots,t \}$ такую, что все $R_i \in R$ находятся в УТНФ, и для каждого экземпляра S и R действительно:

$$\left(\bigcup_{i=1}^{t} F_i \right)^+ = F^+ , \quad S = \underset{i=1}{\overset{t}{*}} R_i , \quad R_i = S \left[U_i \right] .$$

Б/ Процедура повторного проектирования логической структуры уже действующей базы данных.

Повторное проектирование вызывается необходимостью учесть изменение наших представлений об исследуемом объекте. Здесь необходимо в максимальной степени использовать опыт первоначального проектирования, а также принять во внимание опыт эксплуатации базы данных с целью сохранить, где это только возможно, существующую структуру данных. Основная задача повторного проектирования формулируется следующим образом:

Пусть $F=\{f_1, \ldots, f_n\}$ является первоначальным множеством функциональных зависимостей, на основе которых была определена реляционная схема $R=\{R_i\ (U_i, F_i) \mid i = 1, \ldots, t\}$. Требуется установить, повлияют ли изменения в множестве F, вызванные изменением наших представлений и знаний об исследуемом объекте, на логическую структуру базы данных, определенную реляционной схемой R. Если да, тогда необходимо найти новую реляционную схему R^{new}. Рассмотрим три вида изменений:

1. Пусть первоначальное множество функциональных зависимостей $F = \{f_1, \ldots, f_n\}$, множество новых функциональных зависимостей обозначим $F^{new} = \{f_1^{new}, \ldots f_m^{new}\}$. Тогда результирующее множество зависимостей будет равно $F' = F + F^{new}$.

2. Пусть снова первоначальным множеством зависимостей является $F = \{f_1, \ldots, f_n\}$, а множество функциональных зависимостей, которые потеряли смысл и которые следует удалить из множества ограничений целостности, обозначим $F^{del} = \{f_1^d, \ldots, f_k^d\}$. Тогда изменившееся множество функциональных зависимостей будет $F'' = F - F^{del}$.

3. Обозначим снова первоначальное множество зависимостей $F = \{f_1, \ldots, f_n\}$, множество новых зависимостей $F^{new} = \{f_1^{new}, \ldots, f_m^{new}\}$, а множество зависимостей, потерявших смысл, $F^{del} = \{f_1^d, \ldots, f_k^d\}$. Тогда результирующее множество зависимостей будет $F''' = F + F^{new} - F^{del}$.

В дальнейшем изложении будем придерживаться только что введенного членения или разбивки.

Главные функции комплексного алгоритма одинаковы как в основной процедуре, так и в процедуре повторного проектирования. Их можно свести к пяти укрупненным шагам:

1. Удаление из заданного множества функциональных зависимостей $F=\{f_i \mid i= 1,\ldots,n\}$ так называемых редундантных зависимостей, включая удаление так наз. избыточных атрибутов из оставшихся зависимостей, т.е. образование базиса
   $H_1 =\{f_i \mid i = 1,\ldots, n_1\}$,
   $H_1^+ =F^+$.

2. Разбивка базиса на группы функциональных зависимостей с одинаковыми левыми сторонами, т.е. образование базиса
   $H_2 =\{f_i \mid i = 1,\ldots, n_2\}, H_2^+ = H_1^+ = F^+$.

3. Выявление эквивалентных левых сторон, слияние групп с эквивалентными левыми сторонами и образование реляционной схемы в форме $R =\{R_i (U_i, K_i)\ i = 1,\ldots, n_3\}$,
   $U= \bigcup_{i=1}^{n_3} U_i$,
   $\bigcup_{i=1}^{n_3} \{ K \rightarrow U_i - K \mid$
   $K\in K_i$, $K_i \subset U_i$ и $K \rightarrow U_i \in H_2^+\})^+ = H_2^+$;
   где R_i - синтезированные схемы отношений,
   K_i - множества синтезированных ключей этих отношений.

4. Проверка свойства соединимости без потерь.
   Ссылаясь на [7], проверяется с этой целью наличие в R хотя бы одного отношения такого, что действительно $K \rightarrow U \in F^+$. Если в R нет такого отношения, добавляется новое отношение, которое удовлетворяет этому условию.

5. Исключение так называемых излишних атрибутов из R.
   Формальное определение излишних атрибутов дано в [7].
   Неформально атрибут B является излишним в R_i, если он обновимый и несущественный в R_i.

Соответственно операция исключения состоит из двух частей:

- выявление $R_i \in R$ обновимых атрибутов (неформально обновимые атрибуты те, значение которых не требуются для определения других атрибутов),
- тестирование обновимых атрибутов с точки зрения их существенности в R_i. Удаление всех обновимых атрибутов, которые в то же самое время являются несущественными.

В основной процедуре все описанные шаги осуществляются в полном объеме. В процедуре повторного проектирования объем необходимых действий определяется в первую очередь соотношением между множеством функциональных зависимостей, определенных для действующей базы данных, и так называемым результирующим множеством функциональных зависимостей, определенных в соответствии с тремя случаями, упомянутыми в описании задачи повторного проектирования (см. рис. 1-4). Подробное описание соответствующих алгоритмов дается в [11, 12].

Рис. 1 Алгоритм, определяющий объем вычислений при повторном проектировании (АЛГ1)

```
begin   FRES = F + F^new = F^del
        if FRES = H1 then
    go to КОНЕЦ (не требуется изменять R)
    else if  FRES ⊃ H1 then
                проведи повторное проектирование в сокращенном объеме в соответствии с (АЛГ2)
         else if  FRES ⊂ H1 then
                H1 = FRES и продолжай проектирования со второго шага
              else if H1 ∩ FRES  пустое
                      проектируй в соответствии с (АЛГ3)
                   else  проектируй в соответствии с (АЛГ4)

        КОНЕЦ:
end
```

Рис. 2 Алгоритм для определения хода вычислений в случае FRES $\supset$ H1 (АЛГ2)

```
begin
   H1´= H1
      for j = 1 to |F^new| do
        if f_j ∉ (H1´)^+ then
           H1´ = H1´∪f_j
      end
      if H1´- H1 ≠ 0 then
       for i = 1 to |H1| do
         if f_i ∈ (H1´- f_i)^+ then
            H1´ = H1´- f_i
       end
      else go to КОНЕЦ /рис.1/
      endif
end
```

Рис. 3 Алгоритм для определения хода вычислений в случае H1∩FRES = 0 (АЛГ3)

```
begin
   H = H1
   comment для каждой f_i ∈ H1
        for i = 1 to |H1| do
          if f_i ∉ (F´´)^+ then
             H = H - f_i
        end
        H*= H∪F´´
        comment для каждой f_j ∈ F´´
             for j = 1 to |F´´| do
               if f_j ∈ (H*)^+ then
                  H* = H* - f_j
             end
end
```

Рис. 4 Алгоритм для определения хода вычислений в случае $H1 \cap FRES \neq 0$ (АЛГ4)

```
begin
   H = H1 - F''
   comment для всех f_i ∈ (H1 - F'')

   for i = 1 to |H1 - F''| do
     if f_i ∉ (F'')^+ then
        H = H - {f_i}
   end
   H* = H ∪ F''
   comment для всех f_i ∈ H*
   for j = 1 to |H*| do
     if f_j ∈ (H*)^+ then
        H* = H* - {f_j}
   end
end
```

Примечания

1. В алгоритмах АЛГ1 - АЛГ4 индексом j обозначены функциональные зависимости, принадлежащие F^{new}, а индексом i зависимости, принадлежащие $H1$.

2. Для записи алгоритмов использован упрощенный язык высокого уровня, заимствованый из [1].

Завершая обсуждение комплексного алгоритма, следует отметить, что этот алгоритм позволяет получать реляционную схему базы данных с переменной структурой в УТНФ, которая содержит оптимальное количество отношений. При повторном проектировании в максимальной степени сохраняется первоначально спроектированная и созданная структура данных, в результате чего достигаются минимальные потери человеческого труда и машинного времени при реструктурализации базы данных (подробный анализ см. в [12]).

ЗАКЛЮЧЕНИЕ

Предлагаемый комплексный алгоритм должен стать составной частью интегрированной системы для поддержки анализа и обработки экспериментальных данных, которую предполагается постепенно создавать в Центральном вычислительном институте ЧСАН в сотрудничестве с Институтом физиологических регуляций ЧСАН.

ЛИТЕРАТУРА

1. Aho, A.V., Hopcroft, J.E., Ullman, J.D.:
Построение и анализ вычислительных алгоритмов (перевод с англ. The Design and Analysis of Computer Algorithms). Москва, Мир, 1979, 536 стр.

2. Beeri, S., Bernstein, P.A., Goodman, K.:
A Sophisticate's Introduction to Database Normalization Theory.
In: Proceedings of 4-th International Conference on Very Large Data Bases, West Berlin, September 1978, pp. 113-124

3. Beeri, S., Bernstein, P.A.: Computational Problems Related to the Design of Normal Form Relational Schemas. ACM Transactions on Database Systems, vol. 4, March 1979, No. 1, pp. 30-59.

4. Bernstein, P.A.: Synthesizing Third Normal Form Relations from Functional Dependencies. ACM Transactions on Database Systems, Vol. 1, Dec. 1976, No. 4, pp. 277-298.

5. Дрибас, В.И.: Реляционные модели баз данных.
Минск, Издательство БГУ, 1982, 181 стр.

6. Fagin, R., Mendelzon, A.O., Ullman, J.D.:
A Simplified Universal Relation Assumption and its Properties. ACM Transactions on Database Systems, Vol. 7, No. 3, Sept. 1982, pp. 343-380

7. Ling, T.W., Tompa, F.W., Kameda, T.: An Improved Third Normal Form for Relational Databases. ACM Transactions on Database Systems, Vol. 6, No. 2, June 1981, pp. 329-346.

8. Pankrácová, J.: Dekompozice a syntéza relačních databázových schemat (Декомпозиция и синтез реляционных схем-чешск.). Дипломная работа, защищенная на математико-физическом факультете Карлова университета, Прага, 1985, 102 стр.

9. Ullman, J.D.: Основы систем баз данных (перевод с англ. Principles of Database systems). Москва, Финансы и статистика, 1983, 334 стр.

10. Vítková, G.: K problematice využití databázových systémů v AVV (К проблематике использования систем баз данных в автоматизации научных исследований - чешск.). Výzkumná zpráva, č. V-200 , SVT ČSAV, Praha, říjen 1985, 77 str.

11. Vítková, G.: Navrhování relační databáze s měnící se strukturou (Проектирование реляционной базы данных с переменной структурой - чешск.). Informačné systémy, 15, č. 6, 1986, str. 591-602,

12. Vítková, G.: Návrh databáze pro automatizaci vědeckých výzkumů (Проектирование базы данных для поддержки научных исследований - чешск.). Кандидатская диссертация 132 str., Praha, 1986.

MTA SZTAKI Tanulmányok 203/1988 p.23-32

ИНТЕРАКТИВНАЯ СРЕДА ОБЩЕНИЯ С РЕЛЯЦИОННОЙ СУБД

Гусек, Д.
Центральный вычислительный институт
ЧСАН

1. ВВЕДЕНИЕ

В разных отраслях научных исследований проводятся эксперименты, при которых необходимо измерять непрерывные величины в течение длительного периода времени или такие величины, которые очень быстро меняются. Потому в процессе их дискретизации получается большое количество данных.

Примером может служит эксперимент, касающийся проблематики физиологических регуляций, в течение которого получается обычно около 20 МВ данных. Данные, полученные в рамках эксперимента, можно разделить в разные классы, исходя из их физиологического содержания. Обработка данных, принадлежащих к одному классу, имеет свою специфику как с точки врения методов математической обработки этих данных, так с точки зрения предметной области.

В общем можно сказать, что эффективная обработка этих данных возможна лишь в коллективе состоящем из математиков, статистиков, специалистов по моделировании физиологических систем и т.п. Отсюда видно, что этому коллективу научных работников нужно дать в руки мощное средство для манипуляции этими данными, что бы они могли полностью сосредоточиться на анализ этих данных и моделирование физиологических явлений. Упомянутый инструмент должен содержать средства для эффективного хранения данных, выбор произвольных подмножеств этих данных и последующую загрузку результатов обработки этих данных.

Далее компонентами этой системы являются программные

средства для анализа этих данных и средства для накопления знаний, полученных в процессе анализа этих данных. Имеются в виду знания о методах и программах для обработки данных и знания о данных закономерностях исследуемых явлений.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ К СРЕДСТВАМ ДЛЯ ХРАНЕНИЯ ДАННЫХ

Требования к средствам для хранения данных можно сформулировать следующим образом:

1. необходимость хранения большого количества данных;
2. необходимость хранения так называемых секундарных данных, которые возникают в процессе анализа первичных данных ;
3. необходимость ясной и исчерпывающей документации;
4. возможность актуализации данных на основе новых знаний об этих данных;
5. возможность одновременного доступа для многих пользователей;
6. возможность использования данных существующим программным обеспечением для анализа данных на ЭВМ ЕС;
7. способность репрезентации отношений между данными;
8. необходимость средств для защиты данных от их уничтожения;
9. необходимость средств для реструктурализации данных;
10. необходимость непосредственного выбора и загрузки данных в реальном времени с помощью непроцедурального языка.

В большинстве случаев всем этим требованиям отвечают существующие системы управления базами данных (далее СУБД). Требованиям в пунктах /4, 9 и 10/ наиболее соответствуют релядионные СУБД. Однако для эффективного использования СУБД требуются специальные знания, которые касаются методов проектирования баз данных, их образования и эксплуатации.

С другой стороны наибольшую пользу СУБД принесет в том случае, если непосредственно сам исследователь имеет возможность коммуникации с базой данных. Больше всего для таких целей подходит непроцедуральный язык общения с базой данных. Было бы очень полезно, чтобы такой непроцедуральный язык обладал большой силой, чтобы позволял формулировать больший класс

запросов в зависимости от логической структуры базы данных.

Последней важной проблемой является взаимосвязь выбранных данных и прикладных программ для их обработки на физическом уровне.

Решение некоторых упомянутых здесь проблем видим в использовании СУБД реляционного типа в единой среде TSO, позволяющей интерактивную манипуляцию с данными.

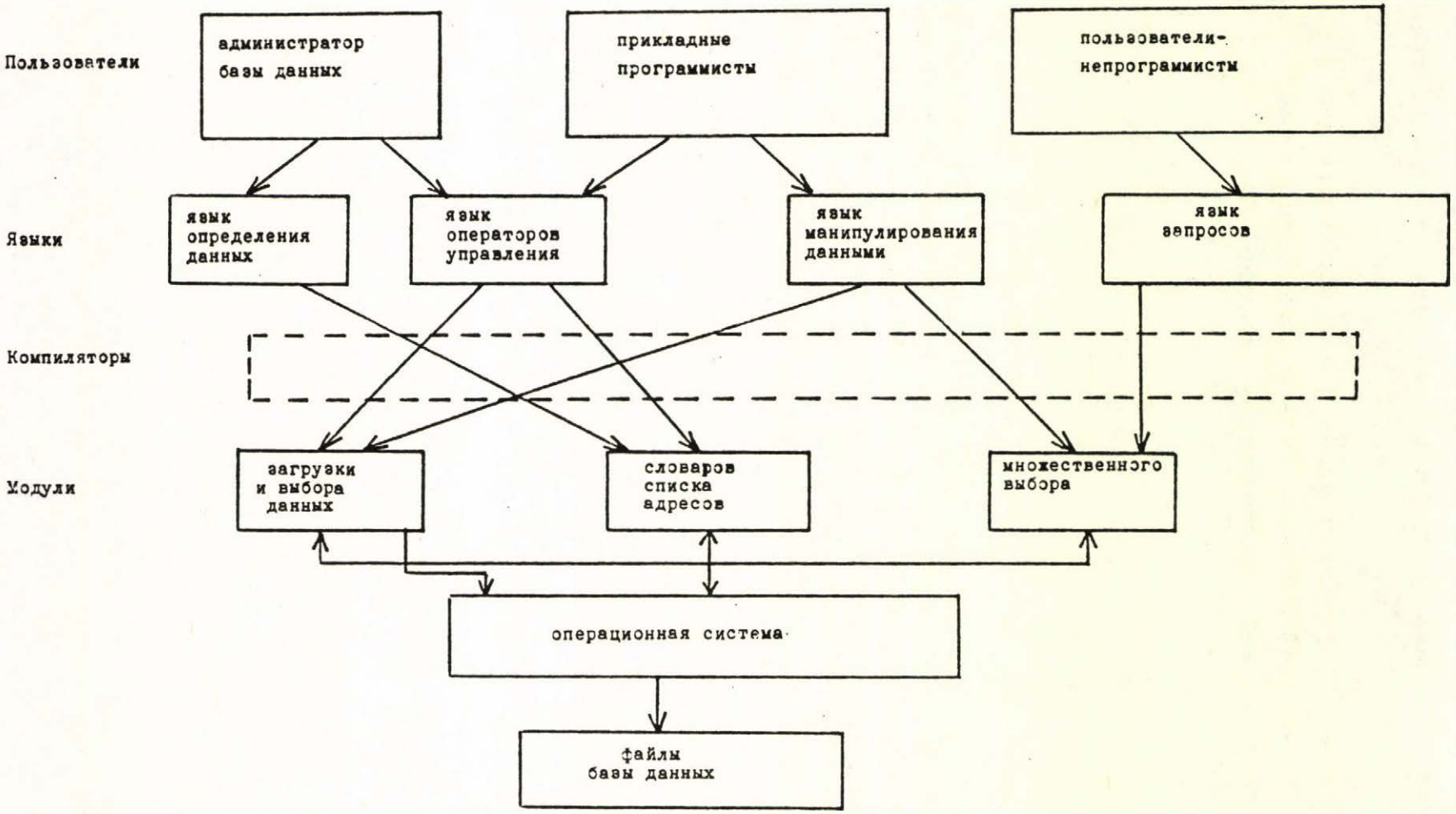
3. СИСТЕМА УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ РЕЛЯЦИОННОГО ТИПА

СУБД реляционного типа SOFIS (Software for Information System) была разработана в Чехословакии в братиславском НИИ социально-экономической информации и автоматизации в управлении (VÚSEI-AR Bratislava) в среде вычислительной машины CDC 3 300.

В настоящее время эта система используется на ЭВМ ЕС и на ЭВМ типа IBM 360/370, которые работают под операционной системой OS.

Развитие этой системы продолжается, в первую очередь это касается разработки телекоммуникационного монитора. Необходимый объем оперативной памяти для пользования этой СУБД составляет около 256 KB. База данных под СУБД SOFIS состоит из семейства файлов, соответствующих отношениям, словаря и списка адресов данных. Последние две составляющие словарь и список адресов данных образуют так называемый интегрированный словарь и список адресов данных SAD и его копия DSAD.

Файлы SAD и DSAD содержит список адресов данных и словарь данных. Поэтому говорим о так называемым интегрированном


Пользователи
администратор базы данных
прикладные программисты
пользователи-непрограммисты
Языки
язык определения данных
язык операторов управления
язык манипулирования данными
язык запросов
Компиляторы
Модули
загрузки и выбора данных
словаров списка адресов
множественного выбора
операционная система
файлы базы данных

СУБД-SOFIS

словаре и списке данных SAD и дупликате DSAD.
На логическом уровне определение этих файлов проводится примерно в том-же объеме, как и образование структуры данных на языке COBOL . С помощью средств операционной системы можно создавать файлы последовательные, индексно-последовательные или файлы с прямым доступом.

У индексно-последовательной структуры можно пользоваться секундарным ключем для доступа к информации. Из файлов с прямым доступом можно получать данные непосредственно с помощью трансформационного алгоритма.

В рамках СУБД различаются три типа файлов: внешние, внутренние и запасные.

Под внутренними файлами подразумеваются файлы, которые были образованы с помощью СУБД и составляют собственную базу данных.

Внешние файлы - это такие файлы, которые были образованы посредством СУБД, но после включения определения этих файлов в SAD и DSAD, они могут обрабатываться как файлы, образованные в среде СУБД.

Запасные файлы содержит копию базы данных и журнальную ленту.

СУБД SCFIS имеет следующие средства для построения базы данных:
- язык определения данных
- язык манипулирования данными
- сервисные программы
- средства защиты данных.

Язык определения данных
Определение данных с помощью языка определения данных в основном заключается в образовании пяти секций в SAD , т.е.:

три обязательные секции: - идентификационная секция
- логическая секция
- физическая секция

и две необязательные: - секция семантики файлов
- секция семантики атрибутов.

Идентификационная секция содержит название файла, пароль файла, названия пользователей и способ доступа: для актуализации или только для выбора данных.

Логическая секция содержит структуру уровней данных, название записи, название атрибутов, включая формат записи в памяти.

Физическая секция содержит идентификацию магнитного носителя файла, метод кодирования, физическую длину записи, избранные структуры хранения данных, название первичного ключа и имена атрибутов, которые образуют этот ключ, название секундарного ключа и имена атрибутов, которые его образуют.

Секция семантики атрибутов и семантики файлов содержит описание файлов или атрибутов на естественном языке, причем слова в апострофах называются ключевыми словами, ими можно пользоваться для получения информации о базе данных.

Язык манипулирования данными работает вместе с языками программирования PL 1 и COBOL . На языке FORTRAN можно пользоваться средствами манипуляционного языка с помощью вызова подпрограмм. Язык манипулирования работает совместно с включающим языком.

Язык запросов RELAN работает в так называемом самостоятельном режиме. RELAN является непроцедуральным, реляционно полным языком. Реляционная полнота этого языка выражается в том, что он позволяет осуществить все операции реляционной алгебры, т.е. объединение, пересечение, разность, декартово произведение отношений, проектирование, деление и фильтрация.

Сервисные программы

Для облегчения работы с базой данных можно пользоваться сервисными программами.
Во-первых, это программа для получения сообщений:

- состоянии базы данных;
- реляционное и статистическое сообщение о базе данных и ее компонентах.

Во-вторых, это программа для актуализации SAD и DSAD. Этой программой можно пользоваться, например, для уничтожения базы данных или файла, удаления имен пользователя из списка имеющих права доступа, включение новых пользователей или секций.

В-третьих, это сервисные программы:
- для расчета памяти на запоминающих устройствах, необходимой для базы данных;
- для загрузки данных в базу данных;
- для образования копии базы данных и ее обратная загрузка ;
- образования секундарных индексов или их уничтожение;
- для акцептирования файла СУБД .

СУБД SOFIS была разработана для пакетной обработки данных.

4. ИНТЕРАКТИВНОЕ РАСШИРЕНИЕ ИСПОЛЬЗОВАНИЯ СУБД SOFIS ДЛЯ ПОДДЕРЖКИ АНАЛИЗА ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ

В Центральном вычислительном институте СУБД SOFIS используется только для поддержки анализа экспериментальных данных в области физиологических исследований, поэтому очень важно, чтобы научный работник смог сам работать с этой системой в реальном времени без потребности создания программ на манипуляционном языке.

Поэтому мы подготовили набор процедур для TSO, которые позволяют работать с базой данных в интерактивном режиме. Для научных работников, которые используют эту систему, очень выгодно что они всю свою работу проводят в единой среде TSO.

Можно было бы разработать для этих целей диалогово-ориентированную систему, но мы решили остановиться на варианте с информирующим файлом (HELP), чтобы не изменять внутреннюю структуру СУБД SOFIS.

Работа с этой системой начинается так, что с помощью вызова процедуры HELP , пользователь получает информацию

о последовательности операций, которые необходимо сделать при создании и эксплуатации базы данных.

Каждую операцию возможно выполнить путем вызова соответствующей процедуры.

Уточненную информацию о пользовании той или иной процедурой можно получить с помощью вызова процедуры HELP с соответствующим параметром. Подробная информация имеет следующую структуру:

Н - название операций
С - синтаксис вызова процедуры для операции
П - значение параметров процедуры
И - принцип использования
В - возможные варианты использования ,

Исключение составляет главный HELP, который вызывается приказом (HELP). Этот HELP содержит последовательность операций для образования и сопровождения базы данных.
В основном эта информация касается:

- определения объема памяти на запоминающих устройствах для файлов и SAD а и DSAD;
- алокации и каталогизации файлов SAD и DSAD;
- определения БД и файлов;
- загрузки;
- использования языка запросов с выходом на дисплей или на АЦПУ и образование файла, хранящего ответ на запрос (так называемый SAVE FILE);
- получения информации об состоянии базы данных;
- интерактивного использования программ на языке манипулирования данными в интерактивном режиме;
- уничтожения БД и файлов.

5. ЗАКЛЮЧЕНИЕ

В заключение можно сказать, что созданные процедуры значительно облегчили коммуникацию с базой данных, позволив работать с ней в интерактивном режиме. Интерактивный режим упрощает и ускоряет образование базы и очень полезен при ее раструктурализации, очень частой операции при использовании базы данных в процессе анализа экспериментальных данных.

Пользование языком запросов в интерактивном режиме не представляет никаких трудностей и для неспециалистов в области систем баз данных. Отсутствие языка для непроцедуральной актуализации приводит к тому, что для актуализации надо привлечь специалиста по базам данных. Не совсем удовлетворительным тоже является ситуация, что отсутствует система для проектирования логической структуры базы данных.

В рамках дальнейшего развития возможностей использования СУБД для упомянутых целей предполагаем включить в общую интегрированную систему для поддержки научных исследований алгоритмы для проектирования логической структуры базы данных, средства для непроцедурального определения данных и инструментарий для совместной работы базы данных и прикладных программ для их обработки.

6. ЛИТЕРАТУРА

1. SOFIS/СУБД: Комплектная документация:
Общее описание системы
Описание языка манипулирования данными
Описание языка запросов RELAN
Сервисные программы
VUSEI-AR, Bratislava 1983-1984

2. Mathé, S. SRBD/SOFIS, Výstavba bázy dát, výber dát a tvorba aplikací
(СУБД/SOFIS, Создание базы данных, выбор данных и использование)
Bratislava, 1986, 198 s.

3. Vítková, G.: K problematice využití databázových systémů v AVV (Výzkumná zpráva č. V-200)SVT ČSAV, Praha 1985
(К проблематике использования систем баз данных в автоматизации научных исследований)

4. Ullman, J.D.: Principles of database systems
Rockville, USA, Computer-Science Press Inc. r. 1980, 489 p.

MTA SZTAKI Tanulmányok 203/1988 p.33-42

МЕТОДОЛОГИЧЕСКИЕ АСПЕКТЫ АНАЛИЗА И ОЦЕНКИ ТЕХНОЛОГИЧЕСКИХ ОБЪЕКТОВ

(Система экспертных расчетов "БИСЕР")

С. Денчев, Д. Христозов, Б. Угарчински

ИЦТТ "ИНФОРМА", 1574 София, ул."Чапаев" 55а, Болгария

Необходимость в преодолении существующего отставания промышленного производства и снижающаяся эффективность реализации его продукции на внешнем рынке требуют, кроме изменений планирования и управления производством и внешней торговли, также и сравнения изделий с аналогичными зарубежными продуктами, технические и экономические параметры которых представляют мировой уровень.

Решающим критерием для экспорта определенного изделия является степень удовлетворения этим изделием требований потребителя при экономически выгодных для него условиях. Вот почему предметом и результатом анализа должны быть изделия с высокой полезной стоимостью, которые могут производиться с возрастающей производительностью труда и относительно низкими расходами. Одновременно с этим изделия должны быть высокоэффективными во время эксплуатации, иметь высокую надежность, легко ремонтироваться и обслуживаться и при этом расход энергии при их функционировании должен быть относительно низким. Для определения этих критериев процесс сравнения изделий предоставляет необходимую информацию, так как на мировом уровне находятся изделия, параметры которых отражают прогрессивные потребительские свойства и в которых соотношение между полезной стоимостью и ценой соответствует условиям внешнего рынка. Определение существующего мирового уровня и его будущего развития позволяет выполнить вариантное определение задания на улучшение существующих изделий и расширения эффективного

производства, соответствующего спросу на международных и внутренних рынках.

Для того, чтобы сравнение изделий могло повысить качество разработки сложных, но реальных планов, оно должно относиться к изделиям, имеющим решающую роль в народном хозяйстве и во внешних экономических отношениях.

Данные, полученные в результате сравнения изделий необходимо включать в общий план долгосрочного развития предприятия или данной отрасли оценивать с этой точки зрения. Прежде, чем окончательно принять определенное заключение, рекомендуется сделать анализ условий на предполагаемых рынках, вместе с количественной характеристикой потребностей и платежеспособного спроса и анализом эффективности предполагаемого производства.

Результаты сравнения имеют прямое отношение к следующим областям инфраструктуры народного хозяйства:

1. Технический прогресс.
2. Управление технологиями.
3. Управление качеством.
4. Формирование цен.
5. Внешняя торговля.
6. Внутренний рынок.

За последние годы в указанных выше областях появились определенные диспропорций, а именно:

- оказалось, что недостаточно выполнить простое, поединичное сопоставление изолированных параметров изделий, для того чтобы направлять процесс научно-технического развития. Развитие невозможно только в рамках экономики отдельной страны;
- было подтверждено, что эффективность управления научно-технического прогресса на уровне производственно-хозяйственной единицы находится под негативном влиянием устарелых методов и форм руководства. Хозяйственные руководители на этом уровне не применяют в достаточном объеме в своей ежедневной практике последние доступные достижения информационных технологий (в частности, богатые документальные и фактографические национальные и международные фонды);
- отчетливо наблюдается различие между "отчетными" показателями качества определенных изделий на экспорт, имеющих

"Большой успех" на внутреннем рынке и их реальное качество, что приводит к неэффективным ценам на международных рынках.

Одним из путей устранения указанных недостатков и диспропорций является использование метода международного сравнения изделий, который находит практическую реализацию в системе экспертных оценок "БИСЕР". Этот метод дает возможность для анализа и коррекции соотношений между ценами, техническим уровнем и качеством изделий на основании учета объективных критериев оценки этих соотношений на внутренних и международных рынках.

"БИСЕР" является программной системой, реализующей метод сравнения изделий, основанный на корреляционном и регрессионном анализе. С помощью аппарата корреляционного и регрессионного анализа определяются:

- соотношение и вид зависимости между техническими параметрами и ценой;
- соотношение и вид зависимости между средней (теоретической) мировой ценой и средним технико-экономическим уровнем.

При сравнении и оценке необходимо использовать, по мере возможности, информацию о всех изделиях с одинаковыми функциями, выпускаемыми высокоразвитыми в промышленном отношении странами. За основу сравнения принимается "средний" мировой уровень основных изделий, выпускаемых самыми крупными и известными фирмами промышленно развитых стран. Для каждого отдельно взятого параметра изделия определяется разница по сравнению с мировым уровнем в виде коэффициента, а взвешенно среднеарифметическое значение этих коэффициентов является сравнительной характеристикой технического уровня сравниваемых изделий. Цена всех изделий задается для одинакового периода и в сопоставимых условиях поставки и оплаты (или выполняются соответствующие коррекции). Теоретическая мировая цена определяется при помощи регрессионной функции. Из функционального соотношения между техническим уровнем и ценой выводится так называемая ориентировочная цена. Эти теоретические цены, конечно, не учитывают торгово-политические и коммерческие факторы, влияющие на реальные цены, согласованные в договоре. Необходимо, однако,

чтобы эти факторы учитывались в качестве определенной целевой ориентации в процедурах и при принятии решений.

Определение мирового технического уровня и теоретической мировой цены при помощи регрессионной функции позволяет сравнивать отдельные свойства изделий и конкретные цены, но не обеспечивает комплексное сравнение изделий. Для этого необходимо сначала привести отдельные технические параметры к сравнительным безразмерным коэффициентам, которые являются характеристикой технического уровня оцениваемого изделия. С их помощи можно сравнивать различные изделия. Изделия, для которых этот коэффициент самый высокий, представляют собой самый высокий уровень, при условии, что для оценки был выбран действительно представительный комплекс изделий. Ориентировочная цена является произведением теоретической /мировой/ цены, полученной при помощи регрессионной функции и сравнительного коэффициента.

Выполненное сравнение предоставляет необходимые данные для планирования и принятия решений, но его статистический характер снижает целесообразность его использования для выработки долгосрочных программ, в которых закладываются будущие мировые технические достижения. Для этой цели необходимо дополнить результаты сравнения с прогнозами для определения тенденций развития решающих параметров; другими словами, необходимо соединить экстраполяцию с определением вероятных временных границ качественных скачков, вызываемых коренным обновлением изделия.

Решающим критерием экспорта определенного изделия является степень удовлетворения потребностей потребителя, таких как: функциональных, технических, экономических, эстетических, организационных и т.д. Поэтому, при сравнивании изделий необходимо рассматривать в комплексе совокупность свойств изделия и искать один или несколько количественных параметров, которые характеризуют его наиболее полно. Такой характеристикой, в большой степени отвечающей этим требованиям, является цена по которой данное изделие может продаваться (конкурентная цена) по сравнению с ценами конкурирующих его изделий. Она находится в прямой зависимости от комплекса функциональных, технических, экономических и др. свойствах изделия. Перед этим сравнением ставятся следующие цели:

1. Выяснение, в какой степени конкурентная цена обеспечивает необходимую прибыль (т.е. является ли разность между конкурентной ценой и производственной ценой достаточно большой)? Можно ли продолжить производство изделия без изменения?

2. Определение какие параметры должны быть улучшены и насколько, чтобы изделие получило конкурентоспособную цену, обеспечивающую минимально необходимую прибыль, с учетом, конечно, и того, что улучшение параметров связано с удорожанием производства.

3. Определить какие параметры могут быть ухудшены и насколько (с целью снижения производственной цены), но так, чтобы это не отразилось в большой степени на конкурентную цену, т.е. в итоге получить увеличение прибыли.

4. Определить значения параметров "среднемирового" и "мирового" изделия.

5. Изучение тенденций и перспективы развития отдельных параметров в ближайшие годы (учитывая, конечно, только количественный рост, и не учитывая качественный рост).

6. Исследование зависимости цены от изменения различных параметров и поиск оптимального, по заданному критерию, соотношения, как в случае импорта так и в случае экспорта данного товара.

Очевидно, базой для сравнения отдельных параметров является "среднемировое" изделие, значения параметров которого получаются как среднее по значениям параметров изделий, входящих в "группу для сравнения" (группа изделий, служащих эталонами в процессе сравнения), т.е. первая задача которая должна быть решена при сравнении изделий, это задача определения состава группы изделий, задающих "среднемировой" и "мировой" уровни. Подбор изделий, включаемых в эту группу имеет существенное значение, как для получаемых оценок, так и для общности выводов, которые могут быть сделаны. Если в группу включены изделия,

конкурирующие на определенном рынке, то и выводы о конкурентоспособности сравниваемого изделия относятся только для этого рынка. При такой постановки решается следующая задача: каким образом на этом рынке выгоднее всего можно продать выпускаемое изделие. Если сравнение делается с целью усовершенствования и улучшения изделия, то в группу должны включится изделия выпускаемые ведущими фирмами в мире, продающихся на большинстве мировых рынков и которые по своим функциональным, техническим и другим качествам опередили остальные изделия.

Вторая задача, это определение номенклатуры параметров, отражающих с наибольшей полнотой свойства изделия. Здесь нужно учитывать тот факт, что основные параметры, определяющие изделие как таковое и выполняющие роль идентификаторов группы изделий не обязательно должны входить в эту номенклатуру.

Чтобы достичь намеченные выше цели, необходимо разработать методы, обеспечивающие необходимую достоверность получаемых результатов при существующих ограничениях процесса сбора первичной информации.

В чем состоит проблема?

1. Обычно, число изделий, естественным образом входящее в группу для сравнения находится в интервале от 5 до 10. Реже до 20 различных объектов обладют нужными качествами чтобы служить эталонами "мирового уровня" или конкурируют на данном рынке. Искусственное увеличивание состава группы для сравнения, путем включения в ней неравностойных объектов приводит к увеличению дисперсии получаемых оценок и в итоге к недостижению намеченных целей.

2. Число параметров, описывающих свойства данного изделия, часто очень велико и выбор наиболее важных, определяющих параметров находится под сильным влиянием субъективных причин. С другой стороны, на конкурентоспособность изделия оказывает влияние небольшое подмножество параметров, часто только один параметр.

3. Обеспечение точных данных о значениях всех необходимых параметров для всех изделий, входящих в "группу для сравнения",

является сложной, часто неразрешимой задачей, в связи с тем, что точные значения некоторых наиболее важных параметров являются производственным секретом и не публикуются.

4. Известные методы математической статистики для получения адекватной оценки функции, зависящей от n аргументов, требуют n^2 наблюдений. В случае же необходим метод , обеспечивающий получения оценки при наличия небольшого числа наблюдений, независимо от числа аргументов.

Несмотря на то, что подготовка входной информации также выполняется при помощи компьютера, функциональные процедуры этой обработки имеют так называемый предпроцессорный характер и они не являются частью математико-статистического аппарата, реализующего сущность метода.

Отдельными укрупненными фазами обработки входной информации при помощи ЭВМ являются:

1. Вычисление корреляционной матрицы.

2. Определение второго (вычисленного) главного параметра.

3. Определение подматриц (технического уровня) для каждого отдельно взятого параметра в зависимости от выбранного экспертами и вычисленного системой главных параметров.

4. Определение технико-экономического уровня изделий в зависимости от выбранного и вычисленного главных параметров. Ранжирование изделий.

5. Выбор модели и определение графических зависимостей между ценой и двумя главными параметрами.

6. Построение таблиц результатов "мировой уровень".

Результаты математико-статистического сравнения технического и технико-экономического уровня изделий, полученные при помощи ЭВМ, должны быть рассмотрены группой экспертов, так как ЭВМ дает объективные, но усредненные значения. Необходимо, чтобы результаты были рассмотрены с учетом более широких взаимосвязей, какими являются, например, конкретные требования определенного потребителя по отношению к техническому или технико-экономическому уровню данного изделия, требования отдельных рынков, окомплектовка изделия производителем и т.д.

Окончательная экспертная оценка получается в двух этапах. На первом этапе, являющимся подготовительной фазой, подготавливается заключительное совещание экспертов. Качество выходных данных прямо пропорционально качеству входных устройств. Из этого основного факта необходимо исходить при оценивании результатов. Оценивание отдельных изделий нужно начинать с оценкой эталона, его позиции в условиях международной конкуренции и его веса в мировой торговле. Определение, в какой степени эталон соответствует фактическому среднему международному уровню, является очень важным, так как коэффициент корреляции показывает, насколько данное изделие по одному или другому специфическому параметру находится ниже или выше этого среднего уровня. Оценка комплексности, актуальности и правильности выбора информации является основой этого определения. Среднее значение для эталона задается как единица.

Между экспертами часто возникают разногласия в связи с выбором главного параметра, и поэтому система определяет второй главный параметр. В результате появляется оценка изделия по нескольким критериям, которые необходимо обобщить путем разработки таблицы всех результатов по отдельным главным параметрам. Благодаря этому (в случае расхождения мнений о значимости отдельных параметров), повышается качество оценки показателей технического и технико-экономического уровня.

Второй этап представляет собой заключительную экспертную оценку результатов сравнения. Группа экспертов должна состоять, по мере возможности, из тех специалистов, которые приняли участие в подготовке входных данных. Группа экспертов сначала знакомится с результатами, после чего оценивает правильность выбранного комплекса изделий, параметров и их оценок. Рассматриваются достигнутые технический и технико-экономический уровни оцениваемого изделия по сравнению со значениями эталонных изделий. Если выяснится, что первичная документация и вычисленные результаты соответствуют опыту и знаниям группы экспертов, можно перейти к объяснению установленных главных отклонений.

При этом можно проверять разные зависимости, существующие на уровне "параметр" между изделиями разных производителей, предлагающих свою продукцию на определенном рынке, или

зависимости, отражающие разные взгляды определенных групп на качественный уровень изделий (например, параметры мощности, эксплуатационные параметры, эргономические параметры и т.д.).По результатам сравнения изделий с наивысшим техническим и технико-экономическим уровнем с изделиями с низким уровнем, можно также судить о наличии определенных тенденций в мировом техническом развитии изделий, о динамике цен на отдельных рынках и т.д.

Существующая программная система позволяет также экспериментировать, главным образом в отношении оценки влияния улучшения определенных технических параметров (или нескольких вариантов такого улучшения) на цену, учитывая уровень конкурирующих друг с другом изделий и фактор времени.

Все заключения, взгляды и опыт, накопленный от проведенных экспериментов, включаются в заключительный отчет об оценивании изделия.

При разработки программной системы экспертных расчетов "БИСЕР" использованы самые современные методы проектирования и программирования больших программных систем, выполненного небольшим колективом. Следуя принципом структурного программирования - технологию "TOP-DOWN" на алгоритмичном языке PASCAL, реализованном для микрокомпьютера типа IBM PC/XT - TURBO PASCAL 3.01A/BORLAND INC., была разработана первая версия системы в течении несколько месяцев.

Система "БИСЕР" состоит из 5 основных модулей, реализующих ее функциональные возможности, архитектуру данных, управление данными, связь с системной средой и интерфейс с потребителем. В составе этих основных модулей входят 63 подпрограммы и функции (PROCEDURE & FUNCTION) описание которых дается ниже.

МОДУЛЬ 1 - ввод, коррекция входных данных и их запись в дисковый файл.

МОДУЛЬ 2 - настройка печати и подбор выходной информации.

МОДУЛЬ 3 - выполнение обработки данных.

МОДУЛЬ 4 - визуализация основных результатов в синтезированном виде на экране монитора.

МОДУЛЬ 5 - управление программной системой.

ЛИТЕРАТУРА

1. Поллард Дж., Справочник по вычислительным методам статистики, Москва,"Финансы и статистика" , 1982

2. Вальтер Кубэ, Фолькмар Лирс, Метод корреляционного анализа и оценки тенденции развития - KORTER, ДИТКХУ, Дрезден, ГДР, 1982

3. Mejinarodni srovnovani vyrobcu metodou "KORTER", UTRIN, 1982

4. TURBO PASCAL version 3.0, Reference manual, BORLAND INC.

MTA SZTAKI Tanulmányok 203/1988 p.43-48

УПРАВЛЕНИЕ ИНФОРМАЦИОННОЙ СРЕДОЙ КАК УСЛОВИЕ ДЛЯ ТРАНСФЕРА ТЕХНОЛОГИЙ

Ст. Димитрова, С. Денчев

ИЦТТ"ИНФОРМА", 1574 София, ул."Чапаев" 55а, Болгария

Обособление информационной индустрии как один из самых важных и динамически развивающихся отраслей промышленного производства, вывело на переднем плане проблемы, связанные как с управлением интерфейсами и коммуникациями между отдельными промышленными и социальными системами, так и с управлением информационными деятельностями, процессами и фондами, составляющими информационный ресурс современного общества.

Одним из основных компонентов информационной индустрии является информационная среда. Информационная среда состоится прежде всего из информационных фондов общего и специального /в разных предметных областях/ предназначения и технологий их обработки.

Исходя из факта, что конвенциональными сегодня являются автоматизированные и компютеризированные /с некоторым резервом/ информационные технологии, целесообразно сделать качественный скачок по отношению к вопросам, отнасящимся к рациональному и эффективному управлению информационной средой, чтобы на завтрашний день эта среда не оказалось тормоза, а основа и катализатор будущего развития промышленного производства и связанных с ним количественных и качественных изменений социального статуса общества.

Понятие "информационная среда" не является глобальным понятием. Пользуясь этим понятием ставим себе цель охватить в одно неделимое целое все информационные фонды и технологии для их обработки. В этом смысле можно говорить о существовании множества информационных сред, в общем случае, определенных конкретными предметно-промышленными областями. Предлагаемая дифференциация, с одной стороны облекчает анализ и управление

деятельностями и процессами внутри информационной среды, а с другой - дает возможность проведения исследований и экспериментов, связанных с использованием методов и средств искусственного интелекта, как основа информационных технологий будущего.

Рассмотрение вопросов связанных с информационным обслуживанием производственного процесса ведет к проблемам научно-технического развития, к повсеместному внедрению достижений научно-технической революции /НТР/.

На настоящем этапе вся хозяйственно-экономическая жизнь руководствуется мнениями и убеждениями о решающем характере науки, о науке как прямая производительная сила, о научно-технических исследованиях - основа планирования и т.д.

Современное развитие НТР является объектом внимания всех уровней хозяйственного и социального управление. Найти правильное направление, точно поставить цели, обеспечить нужные ресурсы и средства, предпосылки и взаимосвязи - все эти проблемы включаются в круг вопросов современного управления и руководства. Выражение этих тенденции могут быть открыты навсюду в мире. Комплексная программа научно-технического прогресса стран-членов СЭВ до 2000 года, множество проблем развития и совершенствования стран европейской экономической общности /ЕИО/ (например ESPRIT, и др.), программы существующие в США, Японии и в других высокоразвитых странах, это доказательство о том внимании, с которым правительства всего мира относятся к развитию науки, техники и технологий. На всех уровнях, совершенно очевидно, что сегодняшнее и будущее развитие науки и техники, это первое и самое важное решающее условие чтобы выжить. Убывание темпа этого развития, это действительный признак замедления в цялостном социально-экономическом процессе.

В связи с поставленными выше проблемами, подчеркивается огромное значение обновления как основа научно-технической и производственной политики на всех уровнях производственно-экономической инфраструктуры общества. Поддерживать высокий градус рыночной активности, добываться этим большого объема

продаж, всегда считалось эталоном правильного и эффективного управления.

На современном этапе развития НТР обновление играет исключительно важную роль в стремлении к высокой рентабельности и эффективности и находится в центре внимания всех производственно-экономических единиц. С целью обновления формируются постоянные и временно действующие звенья для научно-технических исследований как с прикладной, так и с фундаментальной направленностью. Во имя обновления ищут новые научно-технические решения, чтобы внедряя их в практику можно добиться качественно новое состояние экономики.

Вашным условием для проведения научно-технических исследований как с прикладным, так и с фундаментальным характером, позволяющие добиться новых измерений в научно-техническом прогрессе, является наличие актуальной, навременной, точной и дифференцированной информации в разных областях человеческого познания и связанных с ней информационные технологии. Точнее, необходимо иметь информационную среду, анализ и управление которой соответвуют поставленной цели.

Управление сложной системой, такой как информационная среда, базируется на общих принципах и методах теории управления. При этом, каждая конкретная стабильная система имеет собственные, характенные только для нее специфические методы и принципы, которые делают ее устойчивой. В этой связи, информационная среда тоже имеет соответствующую специфику /не будем дискутировать специфические методы и принципы анализа и управления информационной средой. Более углубленно сделаем это в последущих публикациях по этим проблемам/.

Специфику управления информационной средой рассмотрим постольку, поскольку от ее состояния и динамических изменений зависит один из характерных процессов нынешнего развития научно-технического прогресса /НТП/, а именно технологическое обновление.

Один из основных вопросов политики всех высокоразвитых стран, это обеспечение условий, при каторых без особых затруднений можно создавать, трансферировать и использовать

новые технические решения и научные знания. В связи с этим разрабатываются ряд мероприятий для устранения затруднений в реализации иновационной политики. Как показывает опыт один из путей к осуществлению технологического перевооружения, это создание специализированных фондов о технологиях. Технологическая информация, по своей сущности предназначается для формирования оптимальной среды для эффективного трансфера технологий.

Под информационные фонды о технологиях следует понимать базы данных, содержащих документальную и фактографическую информацию о технологиях, примерно в следующем составе:

- область применения;
- описание инженеро-технической сущности:
- технические характеристики используемого оборудования;
- требования к материалам и заготовкам:
- технологические режимы;
- икономические данные:
- точные данные о разработчике;
- библиографическое описание источника информации и др.

Формирование таких информационных фондов, а на их основе и создание баз данных связано с огромными затруднениями, состоящиеся главно в:

1. Накопление и отбор данных.
2. Типизация входных данных с целью последующего сравнительного анализа и оценки.
3. Проектирование и дефинирование баз данных и систем их управления.

При управлении технологической информационной средой особое внимание уделяется следующим вопросам:

- приспособления лексики системы к профессиональному языку конкретных потребителей;
- возможности выбора решении:
- минимизации объема необходимой технологической информации для конкретного потребителя;
- обеспечения средств интерактивной коррекции допущенных потребителем ошибок;

- возможности настройки баз данных и будущих баз знаний к интелекту потребителя.

В последние годы, в связи с наступившими изменениями в социальной инфраструктуре общества и в соответствии с объективными требованиями этапа развития НТР, доминирует мнение, что акт принятия решений не заключается в однократном действии со стороны руководителя, а представляет собой непрерывной цепью управленческих решений, начиная с самого нисшего системообразующего звена.

Высказанное выше мнение, в частности базируется на все более нарастающие информационные потоки. Наступило время когда уже невозможно одному субъекту независимо от его интеллектуальных возможностей, собрать и обработать за предварительно определенное время, необходимую ему информацию. Именно это вызывает необходимость в дифференциации деятельностей при принятии решений, которая основана и связана с дифференциации обработки информации.

Характерно для этой ситуации является то, что все, без исключения, субъекты в управлении должны знать как и каким образом эффективнее использовать информационные источники и каналы, так чтобы обработанная и синтезированная ими информация могла бы послужить на одном из входов системы управления на более высшем уровне.

В последнее время особое внимание уделяется управлению т.н. компютеризированными производствами. Все аспекты производственного процесса там могут рассматриваться в рамках унифицированной автоматизированной системы с единой информационной магистрали. Важным является стандартизирование коммуникаций в процессе управления. Это можно достичь единственным образом с помощью ПРОТОКОЛА ДЛЯ АВТОМАТИЗАЦИИ ПРОИЗВОДСТВА /MAP/. Сегодня характерно отсутствие утвержденных стандартов и протоколов коммуникации между человеком и интеллигентным устройством, а также и между самими устройствами. В области управленческой коммуникации термин ПРОТОКОЛ означает группу правил, которые соблюдаются при передаче и приеме информации.

Использование компютеров в процессе принятия решений в общем случае сводится к созданию т.н. советующих систем. Эти системы предназначены для оказания помощи лицам ответственным в принятии решений на разных уровнях управления, в неструк-турированных или слабоструктурированных ситуациях выбора. Советующие системы в процессе принятия решений, "расширяют" способностей субъектов, но не заменяют их предпочтения к выбору критериев и алтернатив и окончательное их мнение относительно принятия одного или другого решения.

ЛИТЕРАТУРА:

1. Криницкий Н.А., Миронов Г.А., Флоров Г.Д., Автоматизированные информационные системы, Наука, Москва, 1982.
2. 3rd ESPRIT Conference, Proceeding and Press Releases, 29.09 - 01.10, Brussels, 1986.
3. Денчев С.Г.,Левиев Б.Х., Организация и управление на диалога "Човек-компютър", ИЦТТ, София, 1987.

MTA SZTAKI Tanulmányok 203/1988 p.49-55

Управление данных в системе для контроля знаний в сети из микрокомпьютеров

А.Ескенази, Т.Бояджиева
Институт математики с вычислительным центром к Болгарской академии наук

В последные годы разширяется применение микрокомпьютеров в школах. Одна из областей образования, где возможно их использование - это автоматизация контроля знаний. Есть разные способы для оценки и контроля знаний. Удобными для этой цели являются тесты и точнее - обектиные тесты. Из нескольких типов обектиных тестов особенно подходящие для автоматизированной обработки оказываются множественно-выборные тесты. Они легко составляются и проверяются, дают достаточно надежную оценку об уровня знаний. Разсматриваемые системы используют именно этот способ контроля знаний. Для ясности изложения рассмотрим некоторые основные понятия.

Множественно-выборные тесты составлны из тестовых элементов. Тестовым элементом называется вопрос из некоторой предметной области и несколько ответов к нему. Один из ответов правильний, а остальние - неправильние. Задача о автоматизированном контроле знаний при помощи тестов включает нескольких подзадач - о сохранении тестовых элементов, генерировании тестовых элементов, выборке элементов для тестов, сохранении и обработке результатов.

Над этими проблемами уже боле чем десят лет работают в Институте математики к Болгарской академии наук, есть теоретические и практические постижения, опубликованы результаты. За это время реализованы несколько тестовых систем, нашедших применение в учебних заведениях.

Последняя система ТЕСТ, созданная в 1986 году, охватывает все задачи, связаные с автоматизацией множественно-выборных тестов кроме генерирования тестовых элементов. ТЕСТ реализована и успешно аппробирована на 8-битовых микрокомпьютерах типа Правец-82. Использование этих микрокомпьютеров обусловлено тем, что ими оборудованы почти все школьние компьютерные кабинеты в Болгарии.

Предполагается, что предметная область, над которой составлены тестовые элементы, имеет йерархическую структуру и делится на тематические подобласти. Преподавателю предоставлены широкие возможности задавать и поддерживать так называемую йерархическую классификационную схему. Она содержит информацию каким образом учебний материал разбивается на тематические области и подобласти.

Для диалога с преподавателем, работающим с системой, используется язык типа "меню". Он очень удобен и легок для усвоения, поэтому с системой могут быстро освоится и те преподаватели, у которых нет никакой специальной подготовки по использованию компьютеров. Только для подготовки графических изображений, которые могут придружать тестовые элементы, необходимо знакомство с специальным графическим языком и редактором, используемые системой.

Преподаватель может сам создавать и поддерживать тестовая база, где хранятся все тестовые элементы. Каждый тестовый элемент относится к определенному тематическому разделу. Это используется дальше при подготовке тестов. Есть и другие параметры тестовых элементов, которые определяются заранее, например число ответов (от 3 до 5), вес, то есть уровень трудности тестового элемента (от 1 до 3), номер графического изображения, связанного с этим элементом. При подготовке тестовых элементов, кроме русского алфавита можно использовать еще латинский и греческий. Имеются широкие возможности редактирования текста тестовых элементов.

При помощи специализированного графического редактора можно создавать сложные графические изображения, дополняющие тестовые элементы. Таким образом расширяется применение системы и полностью используются графические возможности микрокомпьютера. Графические изображения объединяются в графическую базу.

Перед тестовым экзаменом преподаватель указывает тематические области, из которых будут выбираться элементы для экзамена. Он задает также параметры тестов, которые будут генерироваться - число элементов для конкретного теста, общий вес теста, общее время ответа на поставленные вопросы и так далее. В результате создается так называемый экзаменационный диск, который содержит все необходимые данные, а также и системные программы для экзамена. Его нужно размножить, то есть сделать столько копий, сколько

будет использоваться компьютеров во время экзамена.

В начале экзамена преподаватель указывает учебный предмет, дату и класс. Далее каждый обучаемый вводит свой номер в классе и для него генерируется тест в соответствие с заранее заданными параметрами. Делается случайная выборка из экзаменнационной подбазы, кроме того ответы перемешиваются так, что практически исключено чтобы двое обучаемых получили одинаковые тесты. Вопросы появляются на экране последовательно и обучаемый должен только нажать клавишу с цифрой ответа, которого он считает верным, или специальную клавишу, указывающую, что на заданный вопрос он хочет ответить позже, в конце теста. Вероятность ошибки из-за неумения работы с компьютером очень мала, что освобождает обучаемого из "страха от клавиатуры" и позволяет ему сосредоточить свое внимание на работу с тестом.

Результаты теста автоматически записываются на диске. В конце экзамена все экзаменнационные диски обрабатываются и результаты записываются на специалний диск-журнал. Далее система даем возможность получать разные отчеты и статистические справки о результатах обучаемых. Система уже успешно опробована в одной школе в Софии и будет широко применяться.

К сожалению, наблюдаются некоторые неудобства и ограничения при работе микрокомпьютеров в автономном режиме, в основном из-за небольшого объема внешней памяти компьютеров типа Правец-82. Этот объем не позволяет создавать тестовые базы с более чем несколько сотен тестовых

элементов, что ограничиваем выбор. При этом приходится сохранять файлы на нескольких разных дисках. Например йерархическая классификационная схема и основная тестовая база помещаются на одном диске, а база графических изображений - на другом. Это приводит к неудобствам при работе с системой. Кроме того перед экзаменом преподавателю надо скопировать экзаменационный диск в необходимое число раз, а потом обрабатывать каждый диск чтобы обобщить резульматы.

Конечно, эти проблемы могут решатся с помощью большого компьютера, снабженного терминалами. Но в школах пока невозможно использовать такие компьютеры. Это привело к решению развивать систему ТЕСТ, используя сеть из микрокомпьютеров. Для этого была выбрана сеть ULAN, разработанная в Математическом факултете Софийского университета. Эта сеть пока гомогенна, предназначена для микрокомпьютеров типа Правец-82 и имеет связы между компьютеров типа "шина". К каждому из микрокомпьютеров могут подключаться разные периферийные устройства - дисковые, печатающие и другие. Скорость передачи информации по сети - 200 Кбайтов/сек..

В новом варианте системы (ТЕСТ-М) предусмотрено, чтобы один из компьютеров являлся основным и его можно считать компьютером преподавателя. Существенно то, что у него предполагается емкая внешняя память - твердий диск типа Винчестер, где хранятся все данные тестовой системы. На нем записаны йерархическая классификационная схема, основная

база тестовых элементов, база графических изображений, журнал обучаемых и, конечно, все системные программы. Работа по поддержке йерархической классификационной схемы и основной тестовой базы осуществляется, как и в первом варианте, компьютером преподавателя. Но действия по генерированию теста и обработке результатов в большой степени упрощаются. Перед экзаменом преподаватель также задает тематические области, из которых будут выбираться тестовые элементы. После этого создается таблица тематического выбора, которая содержит номера тестовых элементов, принадлежащих выбранным разделам, и их вес. Перед экзаменом в компьютерах обучаемых загружается программа проведения экзамена. Для каждого обучаемого она содержит таблицу теста с номерами тестовых элементов конкретного теста. Они выбираются в соответствии с заданными в начале преподавателем условиями.

Во время экзамена каждый компьютер считывает по сети связи очередной тестовый элемент прямо из основной базы и, если к нему есть графика – из графической базы. Работается по вопросу оптимизации чтения тестовых элементов из основной базы во время экзамена. С помощью экспериментов будет определен способ группирования выбранных тестовых элементов в специальном буфере, чтобы уменьшить многоразовое чтение из диска.

Результаты обучаемых накапливаются в компьютерах обучаемых, а после окончания экзамена считываются основным компьютером и распределяются в журнале обучаемых на твердом

диске.

Не трудно заметить, что второй вариант системы гораздо удобнее с точки зрения преподавателей. Кроме того, повышается надежность при эксплуатации системы. Мы надеемся, что это обеспечить ее широкое применение в средных болгарских школах, для которых она предназначена.

Исследование и работа по развитию системы ТЕСТ осуществляется при финансовом содействии со стороны Комитета по науке при Совете министров НРБ, договор N 390.

MTA SZTAKI Tanulmányok 203/1988 p.57-64

МЕТОДОЛОГИЯ ПО ПРОЕКТИРОВАНИЮ БД И ВНЕДРЕНИЮ СУБД И ИНФОРМАЦИОННЫХ СИСТЕМ

Румяна К. Киркова

Институт математики с ВЦ - Б А Н
1090 София, П.Я.373

БОЛГАРИЯ

Разработка баз данных /БД/, систем управления базами данных /СУБД/ и информационных систем /ИС/ общего назначения является одним из основных направлений деятельности в области обработки данных. Вместе с потребностями в базах данных в различных организациях непрерывно возрастает число требований на технические и программные средства обслуживания баз данных, обеспечивающие накопление, ведение, выборку и обработку данных.

Для обеспечения гибкости использования данных, что необходимо для их эффективного применения в экономических системах, важными являются два аспекта, которым должны удовлетворять разработки БД.

Во первых, данные должны быть независимы от программ использующих их так, чтобы эти данные могли добавляться или перестраиваться без изменения программ.

Во вторых должна быть обеспечена возможность запрашивать и отыскивать информацию в базе данных без трудоемкого написания программ. Для этого должен использоваться язык запросов данных из базы.

Важнейшей составной частью систем обработки данных и автоматизированных систем управления разными классами является информационная база данных, представляющая собой совокупностью сведений об объекте управления, хранимых в запоминающих устройствах и в программно-технической базе. Программно-технической основой для создания и ведения информационной базы являются ИС и СУБД. Они используются для автоматизации программирования при реализации нескольких часто используемых функций:

- обобщенного описания структуры всех данных, используемых в данной организации,
- создания базы данных,
- поддержки бозы данных, включающей:
  обновление элементов базы данных
  дополнение базы данных
  удаление элементов из базы данных,
- поиска и выдачи информации из базы данных.

Применения, реализованные на основе СУБД и ИС, в принципе являются более сложными, чем обычные применения. Переход на обработку данных с использованием СУБД или ИС определяются принятием решения о целесообразности использования СУБД и/или ИС и проведением следующих аналитических работ:

. оценка целесообразности использования СУБД и ИС,
. оценка, сравнение и выбор конкретной СУБД или ИС из доступных пакетов,
. определение структуры всей базы данных,
. определение трабований прикладных программ,
. определение физической структуры всех данных в базе данных,
. определение состава всех элементов данных и их характеристик

и т.д.

Все это накладывает повышенные требования на некоторые категории пользователей:

- администратор базы данных,
- системные программисты,
- прикладные программисты.

Данным категориям пользователей необходимы глубокие знания относительно объектной системой и ее управления и высокая профессиональная квалификация. Роль администратора БД является очень важной, он должен иметь разностороную квалификацию. Решения администратора БД при определении структуры данных могуть быть определяющими для эффективной работы всей системы. Основная тяжесть при эксплуатации СУБД переносится от прикладных программистов к администратору данных.

Прекладной программист работающий с СУБД или ИС, должен иметь более высокую квалификацию чем обычный программист должен быть знаком с описанием БД в целом или с некоторой его частью, должен знать большинство из возможностей конкретной СУБД или ИС.

Когда пользователи принимают решение о начале работы с СУБД или ИС, они обычно не имеют никакого опыта работы с сложными программными системами. Им необходима методика, следуя которой они могли бы обеспечить правильность выбора СУБД или ИС, составление оптимальной в некотором смысле структуры баз данных, эффективность прикладных программ, организацию внедрения.

Составление методических материалов по СУБД и ИС необходимо рассматривать как длительный, итеративный процесс. С одной стороны, пользователям будут необходимы методические материалы, посвященные вопросам общего характера:

. необходимость и целесообразность применения СУБД и ИС,
. выбор, оценка и сравнение СУБД и ИС,
. оценка применяемости СУБД,
. определение логической структуры БД,
. определение физических характеристик БД

и т.д.

С другой стороны выявляется необходимость в разработке материалов, позволяющих лучше и правильнее использовать возможности конкретной СУБД или ИС, обеспечить организацию работ

по проектированию БД с целью гарантированного внедрения и сопровождения программных систем.

Цель процесса проектирования БД и внедрения и сопровождения программных систем общего назначения состоит в создании информационной базы и комплекса программных средств, обеспечивающих корректное функционирование БД. Под этим в первую очередь понимается удовлетворение информационных потребностей внешных пользователей и обеспечение защиты информации от искажения и разрушений при некоторых некорректных обращениях к БД.

Содержимое одной БД обычно используется различными пользователями в различных целях, поэтому при ее создании должны учитываться точки зрения всех этих пользователей. Проектирование такой БД представляет сложный процесс моделирования информации, который можно в общем виде представить следующим образом:

- анализ пользователя и определение различных информационных его потребностей,
- проблемные пользовательские модели /индивидуальные структуры/,
- интегрирование пользовательских моделей в общую структуру,
- общая структура системы,
- проектирование модель БД,
- логическая структура БД,
- физическая структура БД.

На первом уровне определеяется необходимый состав информации с точки зрения отдельных пользователей, на основании которой строятся проблемные /ориентированные на конкретную работу/ пользовательские модели. Затем, на втором уровне, пользовательские модели объединяются чтобы получить интегрированную, общую, структуру БД, необходимую для всех классов применений. На третьем уровне интегрированная структура данных преобразуется в конкретную структуру, которой СУБД может управлять. И наконец, на четвертом уровне, выполняется физическое проектирование БД. Ограничения, которые различные СУБД и ИС накладывают на этот процесс, имеют место с третьего уровня, т.е. при создании логической и физической структуры БД.

Использование той или иной СУБД предполагает преобразование данных с определенной уже общей структурой в заданные структуры логического уровня, которыми конкретная СУБД может управлять, а затем физическое представление этих данных методами, которыми она располагает. При этом смисловые отношения между данными фиксируются с помощью тех средств, которые представляются конкретной СУБД и в пределах ее возможностей. Отсюда уровень развития СУБД определяется тем, какие возможности они предоставляют для представления данных и смысловых отношений между ними, в результате чего объясняется переход от йерархических структур к структурам типа сетей и далее с реляционным моделям представления данных в БД.

Группа ANSI/X3/SPARC предлагает при проектирования банка данных использовать многоуровневый подход к описанию информационной базы, В,деляя при этом три основных уровней:

- концептуальный
- внутреный
- внешний .

На концептуальном уровне задается формализированное описание информационной базы в терминах конкретной СУБД или ИС. Данное утверждение не относится к общепризнанным, однако нет основания считать его неверным. Во всяком случае, договоримся придерживаться именно этой точки зрения. Отсюда следует, что одна и та же информационная база на концептуальном уровне описывается по разному средствами различных СУБД.

Конечно, что на начальном стадие проектирования администратор БД не может знать какая СУБД удовлетворит требованиям создаваемого банка данных хотя бы поэтому, что эти требования еще не определены. Именно поэтому при проектировании необходимо рассматривать еще один уровень описания информационной базы, на котором можно было бы задать описание информационных сущностей предметной области в терминах, не связанных ни с какой конкретной СУБД. Этот уровень описания информационной базы уже принято называть информационно-логическим или инфологическим. Описание информационной базы на инфологическом уровне указывает что должна содержать и обрабатывать проектируемая система не касаясь пока вопросов как это будет реализовано. Ответ на вопрос как, дается при разработке концептуальной и внутренней схем информационной базы.

Предлагается процесс проектирования разбить на три этапа:

1. ЭТАП разработки инфологической схемы информационной базы, особеность которого состоит в том, что он объединяет совокупность действий, выполняемых на начальной стадии проектирования до привязки к конкретной СУБД,
2. ЭТАП выбора средств реализации, главной задачей которого является выбор СУБД,
3. ЭТАП детализированной разработки банка данных, на котором проектирование выполняется в рамках выбранной СУБД.
   К основным задачам этого этапа относится разработка концептуальной, внутренней и внешней схем информационной базы, а также определение комплекса организационных мер и программных средств, обеспечивающих функционирование банка данных.

Этап разработки инфологической схемы разбивается на следующие три шага:

- определение объектов,
- анализ функциональных связей между объектами,
- определение структурных связей между объектами.

Один и тот же объем может использоваться в одном или нескольких применениях. Но в каждом конкретном применении используется некоторое подмножество атрибутов этого объекта. При определении объектов, задача администратора БД состоит в анализе подмножества реального информационного мира и приложений в интеграций атрибутов, описывающих одну и ту же сущность в единый объект.

Функциональная связь представляет собой элемент алгоритма информационного поиска, при этом она не описывает алгоритм функциональной обработки, а лишь указывает в какой последовательности выбираются экземпляры объектов для этой обработки.

Разработанный первоначальный вариант инфологической схемы может быть подвергнуть преобразованию с целью устранения избыточных объектов и структурных связей. Основное правило преобразований инфологической схемы можно сформулировать следующим образом: из инфологической схемы можно удалить некоторые отдельные связи, если в этой инфологической схеме существуют другие структурные связи, обеспечивающие возможность выполнения всех функциональных связей по которым построены удаляемые структурные связи. При этом могут быть удалены из инфологической схемы и объекти-связки, участвующие только в удаляемых структурных связей.

После построения окончательного варианта инфологической схемы осуществляется выбор СУБД, а затем инфологическая схема отображается в концептуальную в терминах выбранной системы.

Этап детализированной разработки с применением конкретной СУБД или ИС методически обеспечивается технической документацией выбранной программной системы.

Наиболее серьезные трудности возникают при внедрении разработанной системы, ввиду того, что организационной стороне и разработке эксплуатационной документации уделяется недостаточное внимание - усилия сосредотачиваются только на освоении программного обеспечения.

Учитывая опыт создания файловых систем в АСУ, можно предложить следующую организацию работ по созданию информационного обеспечения /ИО/. ИО представляет собой совокупность средств и методов построения информационной базы и подразделяется на внемашинное ИО и внутримашинное ИО.

Внемашинное ИО включает:

- систему классификации и кодирования информации,
- систему нормативно-справочной документации /классификаторы, справочники и нормативные документы/,
- систему оперативной документации,
- систему организации ведения, хранения и внесения изменений в нормативно-справочную документацию.

Внутримашинное ИО включает:
- систему программного обеспечения для создания и ведения информационной базы /ИБ/,
- собственная ИБ, которая представляет собой совокупность специально организованных данных, расположенных на внешних запоминающих устройствах ЭВМ и отображающих объекты системы, отношения между объектами и их характеристик.

Конечной целью работ по проектированию ИО является создание ИБ, методов и средств ее ведения, а также документации технического и рабочего проектов, отражающей все принятые и реализованные проектные решения по ИБ с использованием выбранной СУБД.

Проектирование ИО включает следующие работы:
- сбор и обработку исходных данных для разработки ИО,
- определение списк показателей, используемых в системе,
- разработку системы классификации и кодирования информации,
- разработку форм документов входной, оперативной, нормативно-справочной и выходной информации,
- разработку макетов носителей информации,
- разработку системы ведения внемашинного ИО,
- выбор системы программного обеспечения для создания и ведения ИБ,
- определение логической и физической структуры ИБ,
- выбор средств взаимодействия ИБ с задачами конкретной системы,
- разработку технологии формирования и ведения ИБ,
- проверку проектных решений по ИБ,
- создание и ведение ИБ на конкретном объекте,
- анализ функционирования системы ведения ИБ.

К проблеме эффективного применения современных СУБД можно отметить:
- наличие рядя СУБД различных классов вызывает необходимость выработки технологии выбора системы для конкретного применения,
- обучение на освоение пользователями таких сложных систем требует значительных затрат и длительных сроков,
- использование СУБД при проектировании систем обработки данных требует от проектировщика умения эффективно использовать широкие возможности, предоставляемые системой по реализации информационной базы.

В соответствии с целями и необходимой глубиной освоения системы выделим три уровня мутодических материалов.

Методические материалы первого уровня предназначены для проведения начального циклаизучения системы, результатом которого должно быть принатие решения об использовании или, что не менее важно, отклонении данной конкретной СУБД. К ним относятся:

. краткое, ориентированное на пользователя, описание средств, функциональных возможностей, операционной обстановки, поддерживаемые системой структуры данных,
. описание средств манипулированния данными в СУБД, типовых запросов к системе,
. общие особенности организации и функционирования ИБ средствами СУБД,
. организация работ по проектированию ИБ.

Успешное достижение цели во многом зависит от набора задач, иллюстрирующих основные возможности системы.

Второй уровень методических материалов предназначен для более глубокого освоения пользователям возможностей системы и содержит рекомендации по проектированию ИБ, технологию прикладного программирования в среде данной СУБД. Как подтвердил опыт, наилучшей формой изложения является описание, построенное на базе единой комплексной задачи и содержащее следующие разделы:

- постановка задачи,
- описание алгоритма и соответствующей ему логической структуры,
- конструирование физической базы данных и выбор необходимых логических связей,
- выполнение генерации описаний базы данных,
- разработка программ загрузки,
- разработка программ реализации запросов,
- запуск программ под управлением СУБД.

Методические материалы третьего уровня содержат рекомендации, предотерегающие проектировщика от ошибок, которые приводят к созданию неэффективных, хотя и работоспособных, программ. Они основаны на исследовании типовых, имеющих широкое распространение задач, на моделировании алгоритмов решения, проверке их функционирования на больших реальных массивах, а также на получении аналитических оценок производительности системы в различных ситуациях.

Следуя этой методикой мы подготовили соответствующее описание для одной конкретной задачи - система для автоматизированной поддержки подготовки и проведения научных мероприятий под упровлением ЕС ЭВМ. Эта задача выполняется коллективом из Института математики с ВЦ Болгарской АН и Высшей инженерной школы г.Циттау, ГДР.

ЛИТЕРАТУРА

1. Глушков В.М. - Основы безбумажной информатики, "Наука", Москва, 1982
2. Цикритзис Д., Лоховский Ф. - Модели данных, "Финансы и статистика", Москва, 1985
3. Дейт К. - Введение в системы баз данных, "Наука", Москва, 1980
4. Материалы Первой всесоюзной конференции "Банки данных", Тбилиси, 1980
5. Supper K. - Evaluation of database management systems, Database Journal, Vol.7, N1-1
6. Database systems - Investigation report, Gesellschaft fuer Informatik und Datenverarbeitung, Bonn, 1980

MTA SZTAKI Tanulmányok 203/1988 p.65-70

СПЕЦИФИЧЕСКИЕ ПРОБЛЕМЫ ПРИМЕНЕНИЯ И ВНЕДРЕНИЯ СУРБД

Румяна К. Киркова

Институт математики с ВЦ - Б А Н
1090 София, П.Я.373

БОЛГАРИЯ

ОБЩИЕ ПРИНЦИПЫ

В последнее время большой интерес проявляется с распределенной обработке данных. Этот интерес объясняется в первую очередь тем, что распределенная обработка данных обеспечивает решение актуальных проблем при реализации больших по функциям АСУ, а также АСУ объектами, имеющими территориальную разобщенность.

Под распределенной обработкой данных понимается система из рассредоточенных программных средств и данных на двух или более центрах обработки информации. Эти центры связанны между собой в виде последовательной цепи и/или сети, возможно с общим центром, координирующим работу отдельных центров. Центры обработки информации обмениваются между собой данными через коммуникационные устройства непосредственно или используя интерактивной обмен с устройствами прямого доступа в синхронном режиме. Центры могут обмениватся между собой данными и в асинхронном режиме - такой режим имеет место при последовательном выполнении прикладных программ, т.е. при реализации пакетного режима.

Нетрудно заметит, что ключевым элементом при определении распределенной обработки данных является наличие постоянной информационной взаимосвязи между программными средствами различных центров. Если между двумя или более центрами обработки информации отсутствует информационная связь или данные между этими центрами передаются эпизодически с использованием вспомагательных носителей - как правило с использованием магнитной лентой и/или перфокарт - такие системы принято называть системами децентрализованной /рассредоточенной/ обработки данных.

Главными предпосилками создания систем распределенной обработки данных являются экономические, технические и организационные, так как:

- при построении систем распределенной обработки данных выбор различных типов ЭВМ осуществляется с учетом минимума затрат и максимального повышения производительности систем. Небольшие цепи на мини-ЭВМ, с одной стороны, и снижение затрат при проектировании и эксплуатации систем распределенной обработки данных, с другой стороны, значительно расширяют возможность создания наиболее эффективных систем,

- дальнейшее развитие автоматизированных систем управления предприятием, широта охвата производственных функций в АСУ приводить к тому, что на системы, состоящиеся из одной ЭВМ, накладываются ограничения по мощности и эксплуатации. Очевидно, при реализации больших систем /по функциям или территории/, необходимо распределять обработку и данные на несколько центров,
- создание АСУ на единичной ЭВМ затрудняет оперативную обработку выходных данных т.к. вышислительный центр не может одновременно находится в различных местах возникновения, сбора и выдачи информации. Использование обширной сети удаленных терминалов значительно увеличивает стоймость системы в целом. Создание системы распределенной обработки данных при том же увеличении стоймости приведет к наращиванием дополнительной вычислительной мощности общей системы и позволить приблизить вычислительные средства и данные к месту возникновения, сбора и выдачи информации.

Распределенная обработка требует нового осмысления системы обработки данных как системам централизованной обработки данных высшего порядка.

В чем достойнства распределенных систем перед централизованными системами? Каждый из этих подходов имеет свои достоинства и недостатки, а иногда достоинства одного подхода являются недостатками другого.

Достоинства централизованных систем следующие:
. экономия операций и основных ресурсов системы,
. обобщеный контроль системы,
. простие связи между данными,
. простота актуализации и ввода/вывода данных,
. обеспечение совместимости программных средств и данных.

Достоинства распределенных систем следующие:
. меньшая общая стоймость,
. высокая эффективность системы,
. модульный принцип реализации системы,
. модульный принцип наращивания системы,
. гибкость изменения конфигурации,
. приближение всех ресурсов системы к каждому конкретному пользователю,
. удобство принятия решений.

Распределенная обработка ведет к переосмысление таких понятий, как база данных, администратор базы данных; ставит перед разработчиками навые проблемы; влияет на организацию проектирования систем, эксплуатацию и управлении.

Основные проблемы можно сформулировать следующим образом:

- создание распределенных баз данных,
- разработка систем управления распределенными базами данных,
- организация и управление системами распределенной обработки данных,
- организация связи между центрами обработки информации.

ПРОБЛЕМы ПРОЕКТИРОВАНИЯ СУРБД

Извне /или со стороны пользователя/ распределенная СУБД выглядит как централизованной СУБД, обеспечивающую пользователю возможность описания данных и манипулированния с ними. Внутренная структура СУРБД должна отражать спецификеские условия среды /распределение и копирование данных/, а также и используемую СУБД.

К проектированию СУРБД можно подходить следующими способами:

- методом "сверху-вниз" : при этом вся система проектируется как распределенная с самого начала, т.е. различные функции распределены в отдельных узлах. Результатом применения этого метода являетстя гомогенная система, полученная посредством использования одной СУБД и одного типа данных,
- методом "снизу-вверх" : при этом отдельные функции существующей СУБД интегрированны в одной системе. Этот метод не предоставляет такой свободы проектировщику, как метод "сверху-вниз", а результатом применения этого метода является негомогенная СУРБД,
- методом уровней : это типичный метод проектирования где отдельные уровни системы представляют виртуальные процессоры, которые последовательно обрабатывают отдельные транзакции. Каждому уровню соответствует один "просмотр" базы данных. Самый высший уровень - пользовательский, что соответствует внешней схеме. Следующий уровень - системный, что соответствует глобалной схеме. Распределение базы данных определяется следующими двумя уровнями:
  - уровень декомпозиции - определяет логическую декомпозицию системы /вертикальную или горизонтальную/,
  - уровень распределения - определяет географическое распределение сегментов узлов системы. На этом уровне осуществляется коммуникация в системе,а процессор этого уровня - основный процессор системы. Уровень узла находится под уровнем распределения и связан с транзакциями данного узла. За основу при спецификировании транзакций используется вычисление отношений. Результаты работы транзакций, потверждения или сообщения об ошибках, поступают в обратном направлении, т.е. вверх.

ПРОБЛЕМы РАСПРЕДЕЛЕНИЯ ДАННыХ

Пользователь не обязан знать физическое расположение данных. Он адресует свои транзакции в систему точно также, как если бы она не была распределенной. Этот принцип применяется

чтобы не нарушит целостность данных во время обновления. Если пользователю известно разположение данных, он должен указывать расположение всех копий данных, которые он решил актуализировать, а это угражает целостность базы данных.

Проектировщик распределенных баз данных должен разрешить две специфические проблемы: распределение данных и копирование данных.

Распределение данных: БД описывается своей схемой /характеризирующей используемую модель данных/ и конкретными значениями данных. Известни три способа распределения данных:

- распределения базы данных в соответствии с значениями, т.е. каждый узел включает целостную схему базы, но не все возможные значения отдельных элементов данных,
- распределение базы данных согласно ее структуре, т.е. все возможные значения элементов данного подмножества схемы БД сохраняется в узле,
- комбинация из указанных двух способов.

Согласно теории проблема распределения БД может рассматривать как проблема размещения, которая определяется следующим образом посредством применения классических математических методов. Математическое решение проблемы размещения будет в значительной степени способствовать лучшему пониманию сути распределенной системы, а также показывает, что таким образом нельзя решить вопрос о распределении файлов в распределенной базе данных. Это объясняется тем, что:

- модель пользовательских требований рассматривает доступ к файлу только такого узла /в действительности транзакции, запросы или обновление требует доступа к нескольким файлам нескольких узлов/, т.е. во внимание принимается только стоймость связи между отдельными двумя узлами,
- модель не учитывает коммуникационных расходов на синхронизацию транзакций обновления для отдельных копий данных в системе. Эти расходы могут превышать расходы по самому обновлению,
- модель рассматривает только размещение завершенных файлов, что оказывает влияние совсем недостаточно, посколько во многих ситуациях удобнее применить горизонтальное или вертикалное распределение файла.

Вопреки перечисленным недостаткам, результаты полученные при решения проблемы размещения достаточно ценны, особенно в случае когда пользовательские требования имитируются посредством взаимодействия одним файлом.

Копирование данных: Копирование данных в системе увеличивает устрйчивость системы при появлении ошибок, независимость отдельных узлов и уменшает коммуникационные расходы. С другой стороны копирование данных создает значительные проблемы, связанные со сохранением целостности базы данных во время обновления. Для снятия копий данных имеются следующие три возможности:

- каждый узел содержит копию всей БД,
- в каждом узле сохраняются те части БД, которые не входят в других узлах,
- каждый узел охватывает часть БД - эти отдельные части могуть совпадать, т.е. отдельные логические элементы базы сохраняются на нескольких местах.

Первые две возможности типа "все или ничего" кроме указанных преимуществ /максимальная устойчивость при возможных сбоях в данном узле и обновление только на одном месте/ имеют и значительные недостатки. На практике третая возможность является единственное применимое решение: распределенная БД с копированием часте БД, расположенных в различных местах.

ОБЩИЕ ВОПРОСы ВНЕДРЕНИЯ СУРБД

Практика показывает, что процесс внедрения программных продуктов является длительным, трудоемким и трудно управляемым. С точки зрения разработчиков в нем можно выделить следующие этапы:

- выбор объектов внедрения,
- определение степени участия разработчиков во внедрения,
- участие в самом внедрении,
- накопление колличественных характеристик, полученных в результате внедрения,
- анализ результатов,
- формулировка выводов и рекомендаций.

При выборе объекта внедрения разработчиками могут быть поставлены следующие цели:

- небольшой срок внедрения,
- более полная по возможности проверка функции пакета.

Продолжительность внедрения зависит от следующих факторов:

- степень квалификации пользователей,
- быстрота освоения пакета пользователем,
- наличие накопленных и проверенных /достоверных/ данных для создания реальной базы данных.

Последный фактор может оказать наиболее ощутимое влияние на срок завершения внедрения. Поэтому время экспериментирования там, где уже существуют данные на магнитных носителях в большом объеме, может быть значительно меньше чем времени внедрения там, где создание базы данных начинается с нуля. Наличие подготовленных данных однако накладывает некоторые ограничения на выбор структуры базы данных или на процесс загрузки данных.

При выборе объекта экспериментального внедрения необходимо учитывать и степень охвата объекта внедрения соответствующим пакетом. Экспериментальное внедрение будет тем полезное, чем больше функций и возможностей пакета можно проверить в конкретном применении. В этом направлении разработники СУРБД и ИС столкнулись с проблемой, вызванной тем, что реальные применения

с которых можно начать экспериментальное внедрение требует лишь небольшой части функций соответствующей системы.

Степень участия разработчиков во внедрении зависит от конкретного объекта. Можно считать, что участие разработчиков особенно необходимо в процессе проектирования структуры базы данных и в процессе загрузки базы данных. Участие разработчиков может выразиться и в написании нескольких типовых или примерных программ, которые могут быть использованы в дальнейшем при разработке собственных программ пользователя.

Во всех случаях участие на уровне консультаций и обучения пользователей необходимо до тех пор, пока службы сопровождения и учебные центры не будут в состоянии сами выполнять эту работу. Более активное участие разработчиков во внедрении имеет как преимущества, так и некоторые недостатки. Основное преимущество - это ускорение процесса внедрения. Среди недостатков отметим:

- во первых: участие во внедрении отрывает разработчиков от работы по новым задачам и связано с потерей врреми,
- во вторых: участие разработчиков, как правило, снижает активность самых пользователей, которые расчитывая на то что их работа будет выполнена разработчиками, недостаточно быстро осваивают пакет и не в состянии продолжать начатую разработчиками работу на том уровне.

Результаты внедрения необходимым разработчикам, чтобы оценить необходимость дальнейшего совершенствования системы, разработки дополнительных средств и функций, улучшения документаций, разработки принципиально нового программного продукта, прекращения работы по данному продукту и т.п.

Провести качественный анализ результатов внедрения можно лишь тогда, когда выполнено достаточно количественных характеристик на основе нескольких внедрений, которые осуществлялись на протяжении нескольких лет. Чтобы добиться того, необходимо работу по анализу вести непрерывно. Данный тип работы ведется на разных этапах каждой разработки, пока на одно из внедрений не достигло такого уровня, чтобы можно было сделать достаточно правильные выводы о применимости той или иной системы.

Наконец, существуют применения, в которых работа начата по внедрению, но большинство проводимых работ проделаны только экспериментальными данными. Хотя такие внедрения не дают полной картины применения, информация о них может быть полезной для будущих пользователей.

ЛИТЕРАТУРА

1. Протоколы и методы управления в сетях передачи данных, под ред. Ф.Ф.Куо - "Радио и связь", Москва, 1985
2. Якубайтис Э.А., Архитектура вычислительных сетей - "Статистика", Москва, 1980
3. Дейт К., Введение в системы баз данных - "Наука", Москва, 1980

MTA SZTAKI Tanulmányok 203/1988 p.71-75

ЭКСПЕРТНАЯ СИСТЕМА ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ

Ж.Михайлов, Д.Обретенов, Ж.Ангелов
П.Дишлиева, Н.Кирова, В.Кузнецов
БСНИПИ "Интерпрограмма", София

1. Введение

Проектирование баз данных представляет собой длительный и трудоемкий процес. Основным назначением этого процесса является формирование структуры базы данных, которая с одной стороны отражает корректно предметную область конкретного применения, а с другой - может быть эффективно реализована на основе наличного аппаратного и программного обеспечения.

Процесс проектирования является слабо структурированным. В нем возможно итеративно выполнять отдельные шаги процесса для уточнения результатов. Каждый шаг процесса проектирования характеризуется набором процедур проектирования и критериев оценки результатов. Процедуры проектирования могут быть эвристического или алгоритмического характера. Критерии оценки служат для осуществления оптимального выбора между альтернативными проектными решениями.

В результате процесса проектирования получаются спецификации структуры базы данных (логическая и физическа схемы) и указания для прикладных программистов. Эти спецификации служат для создания базы данных и прикладных программ.

В общем процесс проектирования является сложным и творческим процессом, который имеет разные аспекты: методология проектирования [1,2,3], множество уровней абстракций данных, аналитические средства поддержки проектирования и анализа структур баз данных и средства документирования отдельных этапов этого процесса.

2. Назначение и характеристики системы

Разрабатываемая экспертная система предназначена для проектирования реляционных баз данных. Она представляет собой интегрированное средство [4,5,6,7], которое будет применяться на всех фазах процесса проектирования. Система отличается следующими характеристиками:

а) предоставление базы знаний, в которой включены необходимые результаты из области семантических моделей данных и из области реляционной теории - концепции и алгоритмы. База знаний также содержит эвристические правила для проектирования баз данных и должна отображать конкретную предметную область, для которой создается база даных;

б) предоставление интерактивной метдологической среды, которая позволяет осуществлять отдельные шаги процесса проектирования на основе накопленной информаций;

в) определение общих или специфических принципов логического вывода для конкретной области и создание моделей и правил, на которых они основаны;

г) предоставление возможности включения новых теоретических результатов и правил, так как система является открытой;

д) предоставление развитого семантического интерфейса для конечных пользователей.

3. Необходимые экспертные знания

Знания, необходимые для создания и применения экспертной системы, обособлены в следующие три категории:

- теоретические знания;
- знания, специфические для предметной области;
- знания, специфические для конкретного применения.

Теоретические знания включают концепции проектирования (модели, правила и факты) и методологию проектирования [1,2,3]. Таким образом теоретические знания составлены из алгоритмов, правил и эвристик.

Алгоритмы для проектирования баз данных включают, например, алгоритмы нормализации отношений, алгоритмы отимизации путей доступа.

К правилам относятся техники трансформаций схем, аксиоматика функциональных и многозначных зависимостей, ограничения целостности.

Эвристики включают экспериментальные заключения, статистические оценки, гипотезы о предметной области, упрощение зависимостей между атрибутами и ограничениями.

Для каждой предметной области, для которой создается база данных, существует общая терминология, правила управления, ограничения. Все они представляют собой специфические знания для конкретной предметной области. Эти знания могут быть представлены общими правилами или общими заранее определенными структурами, описывающими часто встречаемые ситуации в данной предметной области. Общие правила и ограничения позволяют лучше отразить семантику определенных объектов или отношений.

Знания, специфические для применения, представляют собой спецификации конкретной прикладной системы. Сюда включаются требования пользователей относительно ожидаемых результатов, функции обработки, входные и выходные данные. На основе этих знаний идентифицируются множество данных и множество операций над ними. Знания данной категории выражаются фактами и правилами, которые составляют детальную спецификацию прикладной системы.

Для представления рассмотренных категорий знаний в системе будут использоваться две разные модели:

- семантическая сеть для представления фактов;
- продукционные правила для представления процесса проектирования и ограничений применений.

4. Методология проектирования

Для осуществления проектирования базы данных и применения средств автоматизации этого процесса необходимо определить соответствующую методологию проектирования. Данная методология рассматривает процесс проектирования как последовательный переход через определенное число уровней абстракции. Процесс проектирования состоит из четырех фаз:

- анализ требований;
- концептуальное проектирование;
- логическое проектирование;
- физическое проектирование.

Первая фаза процесса проектирования представляет собой анализ предметной области, для которой необходимо

создать базу данных. На основе информационных требований и требований обработки определяются объекты предметной области и связи между ними, как и операции, которые выполняются над ними. В результате этой фазы составляется формальная спецификация прикладной системы.

В фазе концептуального проектирования создается концептуальная схема, сохраняемая как семантическая сеть, и связанные с ней ограничения. Отдельные взгляды пользователей объединяются в общее описание после устранения и разрешения конфликтов. В дополнение к синтаксическому контролю осуществляется и семантический анализ для верификации концептуальной схемы. Построение концептуальной схемы и устранение возможных несогласованностей осуществляется интерактивно с помощью пользователей.

В фазе логического проектирования выполняется трансформация семантической сети в нормализованную схему отношений и связанные с ней ограничения целостности. При этой трансформации необходимо сохранить семантику описания предметной области, выраженную концептуальной схемой. В начале фазы логического проектирования экспертная система уточняет интерактивно все ограничения (функциональные и многозначные зависимости, кардинальность ассоциаций [10]). После этого осуществляется нормализация отношений на основе зависимостей между атрибутами.

Назначением фазы физического проектирования является определение оптимизированной схемы базы данных. Сюда включаются набор базисных отношений, набор индексов (первичных и вторичных) и физическая организация отношений.

5. Архитектура системы

Экспертная система состоит как и большинство экспертных систем из следующих компонентов:

- база знаний;
- механизм логического вывода;
- внешний интерфейс пользователей.

База знаний экспертной системы содержит правила и факты. Множество правил в базе знаний составлено из правил, описывающих теоретические знания для проектирования баз данных, и правил, описывающих семантику предметной области и конкретное применение. Различаем следующие категории правил:

- правила синтаксического и семантического анализа спецификаций применения;
- правила трансформации от семантической сети до реляционной схемы в четвертой нормальной форме;
- правила интерактивного приобретения знаний;
- правила сервисных функций;
- метаправила, которые управляют последовательностью шагов проектирования.

Правила проектирования могут быть выражены продукционными правилами следующего формата:

IF C1 AND C2 AND ... CN THEN A1 AND A2 AND ... AM

Здесь Ci представляют собой условия, которые должны удовлетворяться, и Ai представляют собой действия, которые должны быть предприняты. Каждое Ci представляет собой предикат, аргументы которого могут быть фактами базы знаний или функциями. Каждое Ai представляет собой элементарную

операцию над базой фактов или составную операцию, выраженную дополнительными правилами.

Множество фактов моделирует конкретную предметную область. Для достижения этой цели следует иметь возможность определения и поддержки структурированных объектов и связей между ними, отражающих объекты и связи в реальном мире. Для представления фактов в базе знаний выбрана специфическая семантическая модель данных - семантическая сеть [11]. Семантическая сеть определяется как множество помечанных узлов и связывающих помеченных ветвей. Данная семантическая сеть содержит большинство из концепций семантических моделей: агрегация, обобщение и классификация. Семантика предметной области определяется классификацией и идентификацией узлов и ветвей, а также определением некоторых ограничений для отдельных узлов и ветвей.

Семантическая сеть может быть определена как : SN (NC,AC,IC), где NC - означает категории узлов, AC - категории ветвей, IC - категории ограничений целостности.

Категории узлов включают: атрибуты, значения, сущности и экземпляры. Здесь атрибуты и значения являются атомарными объектами, а сущности и экземпляры - молекулярными объектами.

Категории ветвей включают следующие семантические связы: агрегация, ассоциация, классификация, обобщение и эквивалентность. Каждая категория ветвей определяется категориями узлов, которые она связывает, ее направлением и предикатом, выражающим ее семантику.

Ограничение является уточнением семантики данного узла или ветви. Ограничения могут быть выражены дополнительными узлами и/или ветвями, или предикатами. Возможные категории ограничений могут быть: ограничения доменов, ограничения функциональных и многозначных зависимостей.

Механизм логического вывода позволяет осуществлять процесс дедукции на основе правил проектирования. Согласно методологии проектирования процесс проектирования разделен на отдельные шаги. На каждом шаге необходимо либо доказать некоторую гипотезу, либо произвести трансформацию от одной формы представления в другую форму, применяя релевантные правила. Это приводит к использованию двух механизмов - механизмы обратной и прямой цепочки рассуждений. Переключение от одного типа рассуждений к другому осуществляется метаправилами. Сама методология проектирования баз данных описывается иерархией метаправил.

Для управления поиском нужных правил применяется объединение правил в блоках. Предусматривается также применение принципа объектно-ориентированного программирования. Это позволит распределить управление правилами между отдельными правилами, блоками правил и объектами из предметной области.

С помощью внешнего интерфейса проектировщик базы данных описывает конкретное применение. Спецификации требований пользователей составляются декларативным языком.

Отдельные промежуточные результаты процесса проектирования могут быть представлены графическим интерфейсом.

6. Средства реализации экспертной системы

Для реализации прототипа экспертной системы будет

использован язык логического программирования PROLOG. Декларативный синтаксис программ на PROLOG и их процедурная семантика позволяют сочетать декларативное и процедурное представление знаний. Использование языка PROLOG таким образом дает возможность создания интегрированного описания структур данных ("фактов") и алгоритмов ("правил"). База знаний будет организована из отдельных модулей. Набор определенных модулей составляет уровень абстракции знаний. Создание уровней абстракции сверху вниз позволяет осуществлять постепенное уточнение знаний.

С другой стороны, однако, язык PROLOG, как средство создания экспертных систем, не имеет более развитых средств представления знаний, позволяющих осуществлять классификацию агрегирование, обобщение и ассоциацию классов объектов, не поддерживает механизма прямой цепочки рассуждений, не проводит объяснения результатов логического вывода ("HOW" и "WHY"). Эти средства будут реализованы при разработке прототипа на PROLOG. Кроме того, будет реализован интерпретатор входного языка описания предметной области.

Реализация прототипа экспертной системы будет осуществляться для микро-ЭВМ типа IBM PC/XT/AT под управлением ОС, совместимой с MS-DOS.

Л И Т Е Р А Т У Р А

1. Olle, T.W. et al, (eds.), "Information System Design Methodologies: A Comparative Review", North-Holland, 1982.
2. Olle, T.W. et al, (eds.), "Information System Design Methodologies: A Feature Analysis", North-Holland, 1983.
3. Olle, T.W. et al, (eds.), "Information System Design Methodologies: Improving the Practice", North-Holland, 1986.
4. Создание концепции комплексной системы управления информационной базой "СУИБ-Концепция" - Итоговый отчет, Интерпрограмма, София, 1986.
5. Furtado, A., C. Moura, "Expert Helpers to Data Based Information Systems", Proc. 1st Int. Workshop Expert Database Systems, 1984.
6. Shepherd, A., L. Karscheberg, "PRISM: A Knowledge - based System for Semantic Integrity Specification and Enforcement in Database Systems", "Proc. ACM Conf. on Management of Data, 1984.
7. Bouzeghoub, M. et al, "Database Design Tools: an Expert System Approach", Proc. 11th Int. Conf. VLDB, 1985.
8. Fagin, R., "The Decomposition Versus the Synthetic Approach to Relational Database Design", Proc. 3rd Int. Conf. on Very Large Data Bases, Tokyo, Oct. 1977, pp. 441-446.
9. Zaniolo, C. and M.A.Melkanoff, "On the Design of Relational Database Schemas". ACM Trans. Database Syst., vol. 6, № 1, 1981, pp. 1-47.
10. Yao, S.B. (ed.), "Principles of databaze Design", Vol. 1, Logical Organizations , Prentice-Hall, 1985.
11. Griffith, R., "Three Principles of Representation for Semantic Networks", ACM Trans. on Database Systems, Vol. 7, № 3, 1982.

MTA SZTAKI Tanulmányok 203/1988 p.77-81

ОБ ИНТЕГРАЦИИ ИНФОРМАЦИОННЫХ СИСТЕМ

Г.Ю.НИККО, Д.ЭЛЬСТНЕР

ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР АКАДЕМИИ НАУК ГДР

ИНФОРМАЦИОННЫЕ СИСТЕМЫ (ИС) ИМЕЮТ - В СВЯЗИ НЕПРЕРЫВНО РАСТУЩИХ ИНФОРМАЦИОННЫХ ПОТРЕБНОСТЕЙ ПОЛЬЗОВАТЕЛЕЙ - ОДНО НЕПРИЯТНОЕ СВОЙСТВО: СУЩЕСТВОВАТЬ ОЧЕНЬ ДОЛГО И РАСТИ ПО КАЧЕСТВУ И КОЛИЧЕСТВУ, ПРИСОЕДИНЯЯ ВСЕ НОВЫЕ И НОВЫЕ КОМПОНЕНТЫ ИЛИ ОБЪЕДИНЯЯСЬ С ТАКИМИ ЖЕ СИСТЕМАМИ.

НАМИ РЕАЛИЗОВАНО НЕСКОЛЬКО ИНФОРМАЦИОННЫХ СИСТЕМ, КОТОРЫЕ МОЖНО РАЗЛИЧАТЬ ПО ДВУМ ТИПАМ.

ТИП I

У ЭТОГО ТИПА ИМЕЕТСЯ

- БОЛЬШОЙ ЗАПРОС
- ИНТЕРАКТИВНАЯ ФОРМА ДОСТУПА К ИНФОРМАЦИИ
- ОТНОСИТЕЛЬНО МАЛО ДАННЫХ ВВОДА
- МАЛО СТУПЕНЕЙ ОБРАБОТКИ ИНФОРМАЦИИ
- НЕ СТРОГИЙ ВРЕМЕННОЙ РЕЖИМ
- ПРЕОБЛАДАНИЕ НЕСТАНДАРТНЫХ ЗАПРОСОВ
- БД РАЗНЫХ СУБД(TOPAS, DBS/R)

ИМЕЮЩИЕСЯ ДАННЫЕ ЭТИХ БАЗ ДАННЫХ СИЛЬНО КОРРЕЛИРОВАНЫ МЕЖДУ СОБОЙ. К ЭТОМУ ТИПУ ОТНОСЯТСЯ ПОЧТИ ВСЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ АКАДЕМИИ НАУК, А ИМЕННО

- ПЕРСОНАЛЬНЫЕ ДАННЫЕ
- СТАТИСТИКА
- ПАТЕНТЫ
- МЕЖДУНАРОДНОЕ СОТРУДНИЧЕСТВО
- СОТРУДНИЧЕСТВО В РАМКАХ СТРАНЫ
- ХОЗРАСЧЕТ
- РАБОЧИЕ СИЛЫ

ТИП II

ХАРАКТЕРИЗУЕТСЯ ТЕМ, ЧТО

- ОБЪЕМ ЗАПРОСА ИЗВЕСТЕН
- ИНТЕРАКТИВНАЯ ФОРМА ДОСТУПА ПОКА НЕРАЗВИТА
- БОЛЬШОЙ ОБЪЕМ ВВОДИМЫХ И ВЫВОДИМЫХ ДАННЫХ
- МНОГО СТУПЕНЕЙ ОБРАБОТКИ ИНФОРМАЦИИ
- СТРОГИЙ ВРЕМЕННОЙ РЕЖИМ ОБРАБОТКИ ДАННЫХ
- СТАНДАРТНЫЕ ФОРМЫ ОБРАБОТКИ ИНФОРМАЦИИ ПРЕОБЛАДАЮТ.

К ЭТОМУ ТИПУ ОТНОСЯТСЯ ВСЕ БИБЛИОГРАФИЧЕСКИЕ БАЗЫ ДАННЫХ.

ОСУЩЕСТВЛЕНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ ПРИВЕДЕТ К ДВУМ ОСНОВНЫМ ПРОБЛЕМАМ :

I. ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ, Т.Е. ИСПОЛЬЗОВАНИЕ ОПРЕДЕЛЕННЫХ ПРОГРАММНЫХ СИСТЕМ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ.

II. РЕАЛИЗАЦИЯ ВЗГЛЯДА ПОЛЬЗОВАТЕЛЯ.

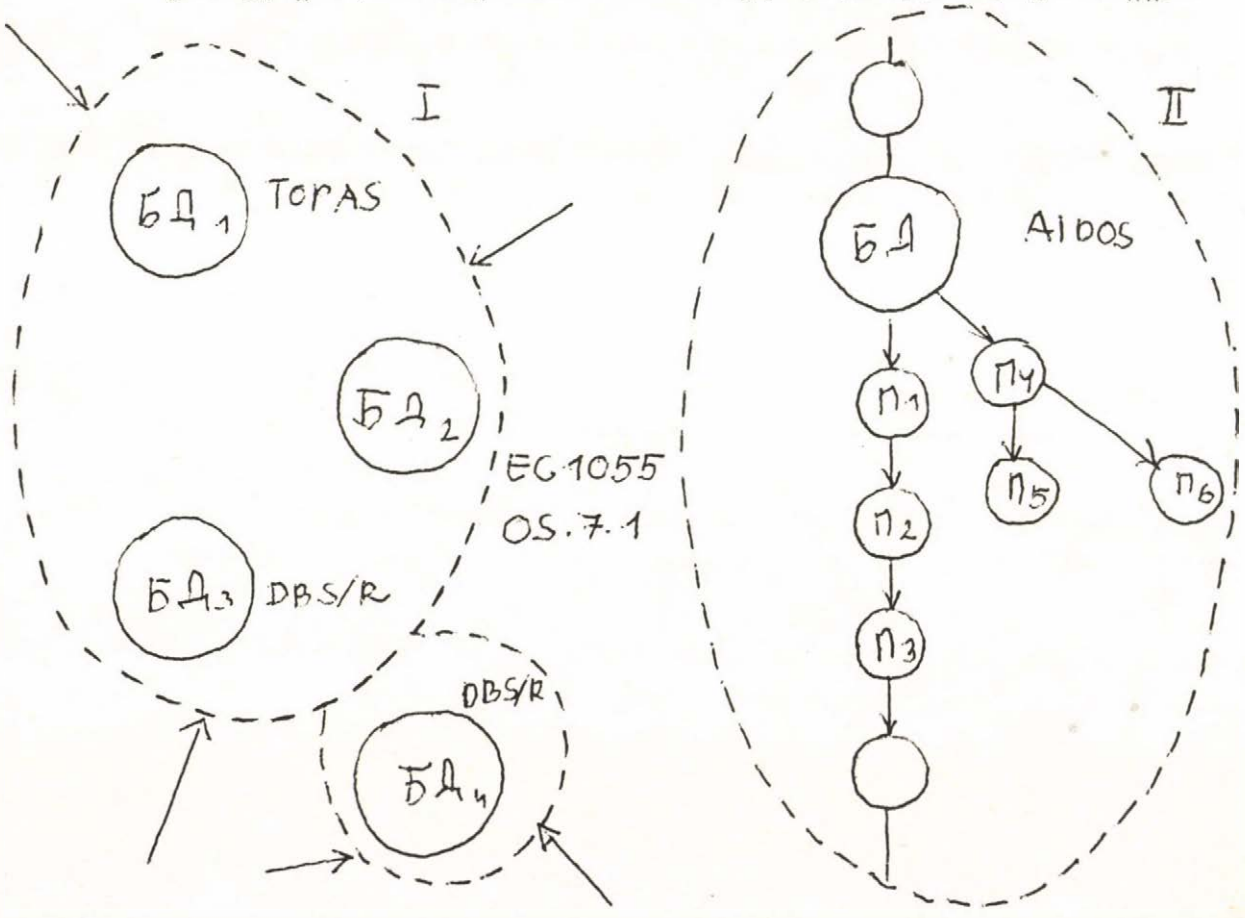
ОКАЗАЛОСЬ, ЧТО ЭТИ ПРОБЛЕМЫ ИМЕЛИ В ДВУХ ТИПАХ СИСТЕМ РАЗНЫЕ ЗНАЧЕНИЯ. ВО ВТОРОМ ТИПЕ ИЗ ЗА ВЫСОКОГО ЧИСЛА СТУПЕНЕЙ ОБРАБОТОК И ПРОГРАММНЫХ КОМПОНЕНТ ПРЕОБЛАДАЮТ ВОПРОСЫ О ТЕХНОЛОГИИ. ИЗ ЭТОГО ПРОБЛЕМНОГО КРУГА РАЗВИВАЛОСЬ ТРЕБОВАНИЕ О ПРОГРАММНЫХ СИСТЕМАХ, КОТОРЫЕ ПОДДЕРЖИВАЮТ СЛОЖНЫЕ ПРОЦЕДУРЫ ОБРАБОТКИ ДАННЫХ.
БЫЛА ПОСТАВЛЕНА ЗАДАЧА О РАЗВИТИИ "АВТОМАТИЗИРОВАННОГО УПРАВЛЕНИЯ ПРОГРАММ" (АУП) .

У СИСТЕМ ПЕРВОГО ТИПА ПРОБЛЕМЫ ВОЗНИКАЮТ ПРЕЖДЕ ВСЕГО ИЗ ЗА ТОГО, ЧТО ДЛЯ УДОВЛЕТВОРЕНИЯ ИНФОРМАЦИОННЫХ НУЖД АДМИНИСТРАЦИИ НУЖНО БЫЛО СОЗДАТЬ НЕСКОЛЬКО ЧАСТНЫХ ИС. ИМЕЕТСЯ НЕСКОЛЬКО БАЗ ДАННЫХ ОТНОСИТЕЛЬНО МАЛОГО ОБЬЕМА, И ПОКА ЧИСЛО СТУПЕНЕЙ ОБРАБОТКИ ИНФОРМАЦИИ НЕ ВЕЛИКО. ПРЕЖДЕ ВСЕГО ВОЗНИКАЮТ ПРОБЛЕМЫ ПРИ РЕАЛИЗАЦИИ ИНФОРМАЦИОННЫХ НУЖД, КОТОРЫЕ ЧАСТО НЕ СТАНДАРТНЫ. ЗДЕСЬ РАНЖИРУЮТ НА ПЕРВОМ МЕСТЕ ПРОБЛЕМЫ, СВЯЗАННЫЕ С ТРАНСФОРМАЦИЕЙ ЗАПРОСОВ ПОЛЬЗОВАТЕЛЕЙ НА РАЗНЫХ ЯЗЫКАХ МАНИПУЛЯЦИИ ИНФОРМАЦИИ И ПРЕДСТАВЛЕНИЕМ ПОЛУЧЕННЫХ ДАННЫХ. ЗДЕСЬ ГЛАВНАЯ ПРОБЛЕМА - СОЗДАНИЕ ЕДИНОГО ВЗГЛЯДА ПОЛЬЗОВАТЕЛЯ ДЛЯ РАЗНЫХ ЧАСТНЫХ -ИС-.

КАРТИНА 1: ИНТЕГРАЦИЯ ИНФОРМАЦИОННЫХ СИСТЕМ ТИПА I И ТИПА II

У ТИПА I - СОЕДИНЯЮТСЯ СИСТЕМЫ НЕСЛОЖНОГО ХАРАКТЕРА.

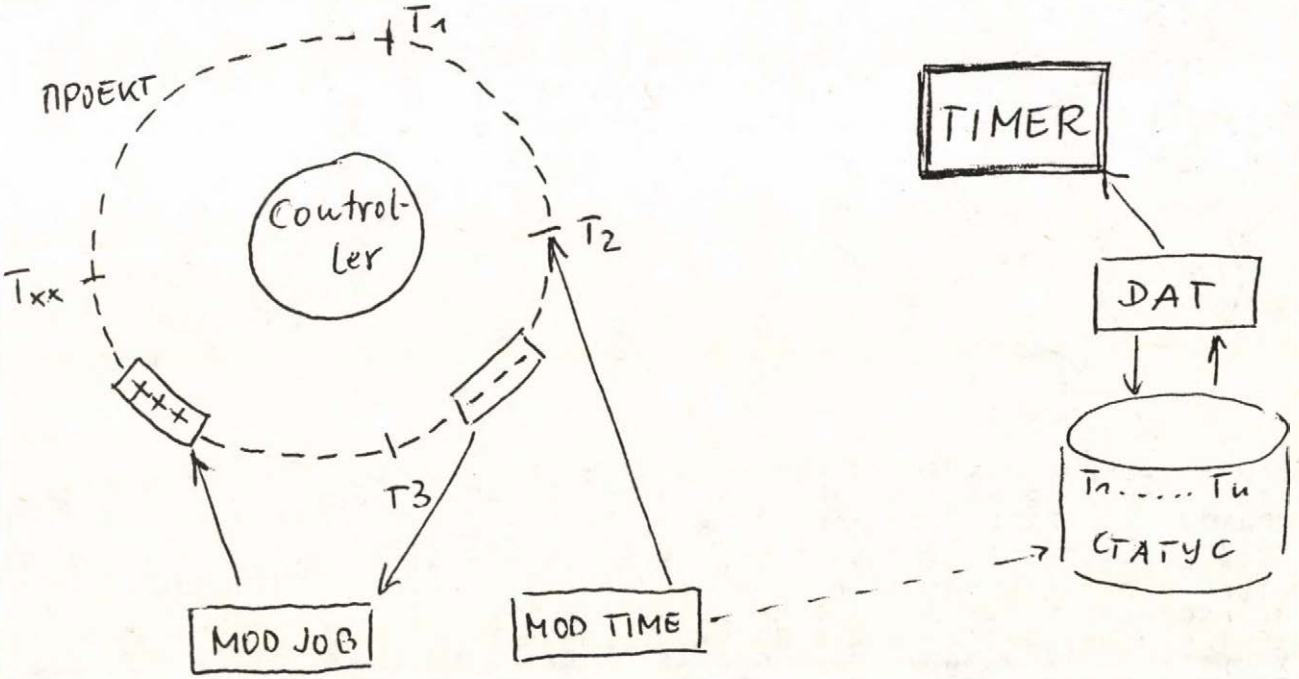
У ТИПА II- ВКЛЮЧАЮТСЯ НОВЫЕ ТЕХНИЧЕСКИЕ И ПРОГРАММНЫЕ КОМПОНЕНТЫ, УСЛОЖНЯЕТСЯ ВСЯ ТЕХНОЛОГИЯ ОБРАБОТКИ ДАННЫХ



ДОВОЛЬНО ДОЛГО РАЗРАБАТЫВАЕТСЯ У НАС ПРОГРАММЫ ДЛЯ -АУП-. ОСНОВНАЯ ИДЕЯ УПРАВЛЕНИЯ СОСТОИТ В ТОМ, ЧТО УПРАВЛЕНЧЕСКИЕ ЗАДАЧИ ВЫПОЛНЯЮТСЯ ТАКЖЕ КАК ОБЫЧНЫЕ ЗАДАЧИ ДЛЯ МАШИНЫ И ПРЕЖДЕ ВСЕГО ДЛЯ УПРАВЛЯЕМЫХ ЗАДАЧ КОНТРОЛЬНЫЙ ЯЗЫК (JCL) НЕ МЕНЯЕТСЯ.

КАРТИНА 2:

ВРЕМЕННОЙ РЕЖИМ ОБРАБОТКИ ИНФОРМАЦИИ В СИСТЕМЕ КОНТРОЛИРУЕТСЯ СПЕЦИАЛЬНОЙ ЗАДАЧЕЙ - "TIMER". УПРАВЛЕНЧЕСКАЯ ЗАДАЧА "CONTROLER" ВЫЗОВЕТ СООТВЕТСТВУЮЩИЕ СЛУЖБЫ -АУП- ДЛЯ МОДИФИКАЦИИ ПАРАМЕТРОВ ОБРАБОТКИ "MOD JOB", ДЛЯ ОБНОВЛЕНИЯ СРОКОВ ОБРАБОТКИ "MOD T", КОТОРЫЕ ХРАНЯЮТСЯ В "STATUS".

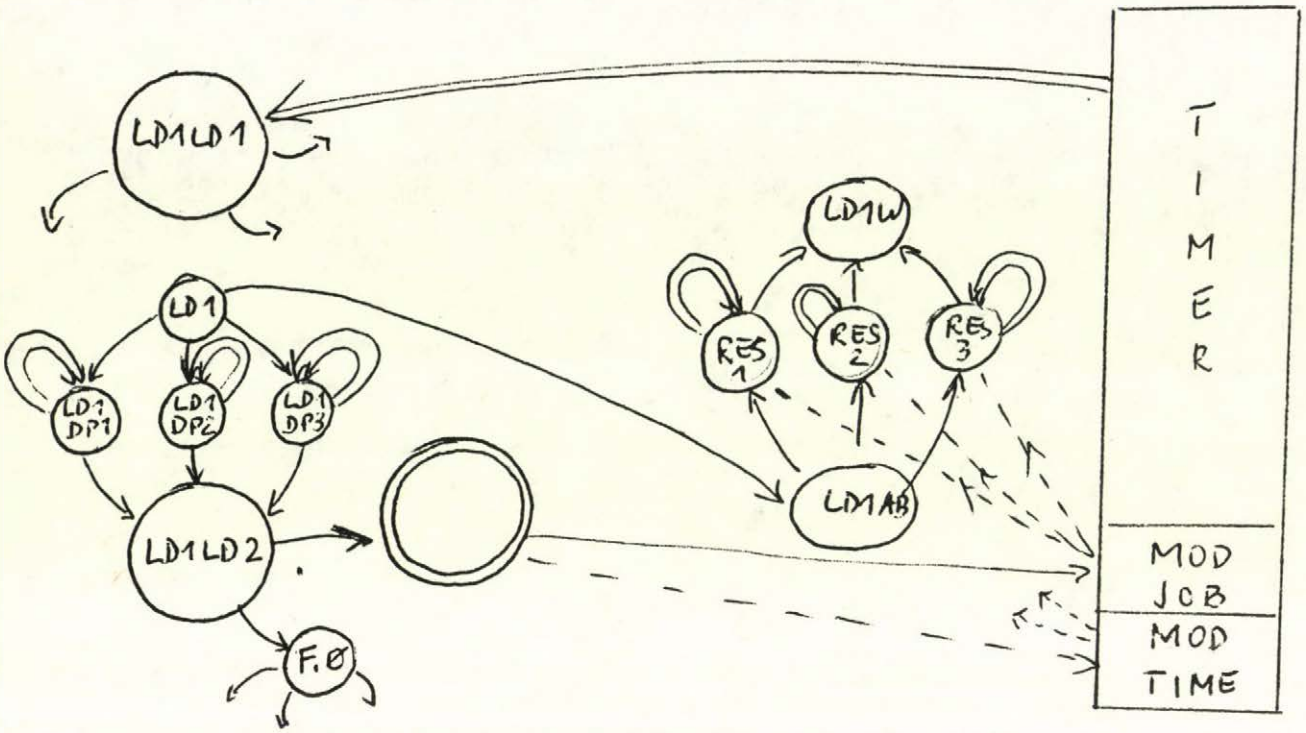


В НАСТОЯЩЕЕ ВРЕМЯ ПЕРВАЯ ВЕРСИЯ ВСЕХ СЛУЖБ ЭТОЙ СИСТЕМЫ РЕАЛИЗОВАНА ОТНОСИТЕЛЬНО САМОГО УПРАВЛЕНИЯ ПРОГРАММ СУЩЕСТУЮТ ДВЕ ВОЗМОЖНОСТИ:

1. ПРОГРАММИРОВАНИЕ ВСЕХ ЛОГИЧЕСКИХ СВЯЗЕЙ.
2. ВТОРИЧНОЕ РАЗЧЛЕНЕНИЕ ОСНОВНЫХ ФУНКЦИЙ УПРАВЛЕНЧЕСКИХ ПРОГРАММ И ИСПОЛЬЗОВАНИЕ НОВЫХ ВОЗМОЖНОСТЕЙ ОПЕРАЦИОННОЙ СИСТЕМЫ.

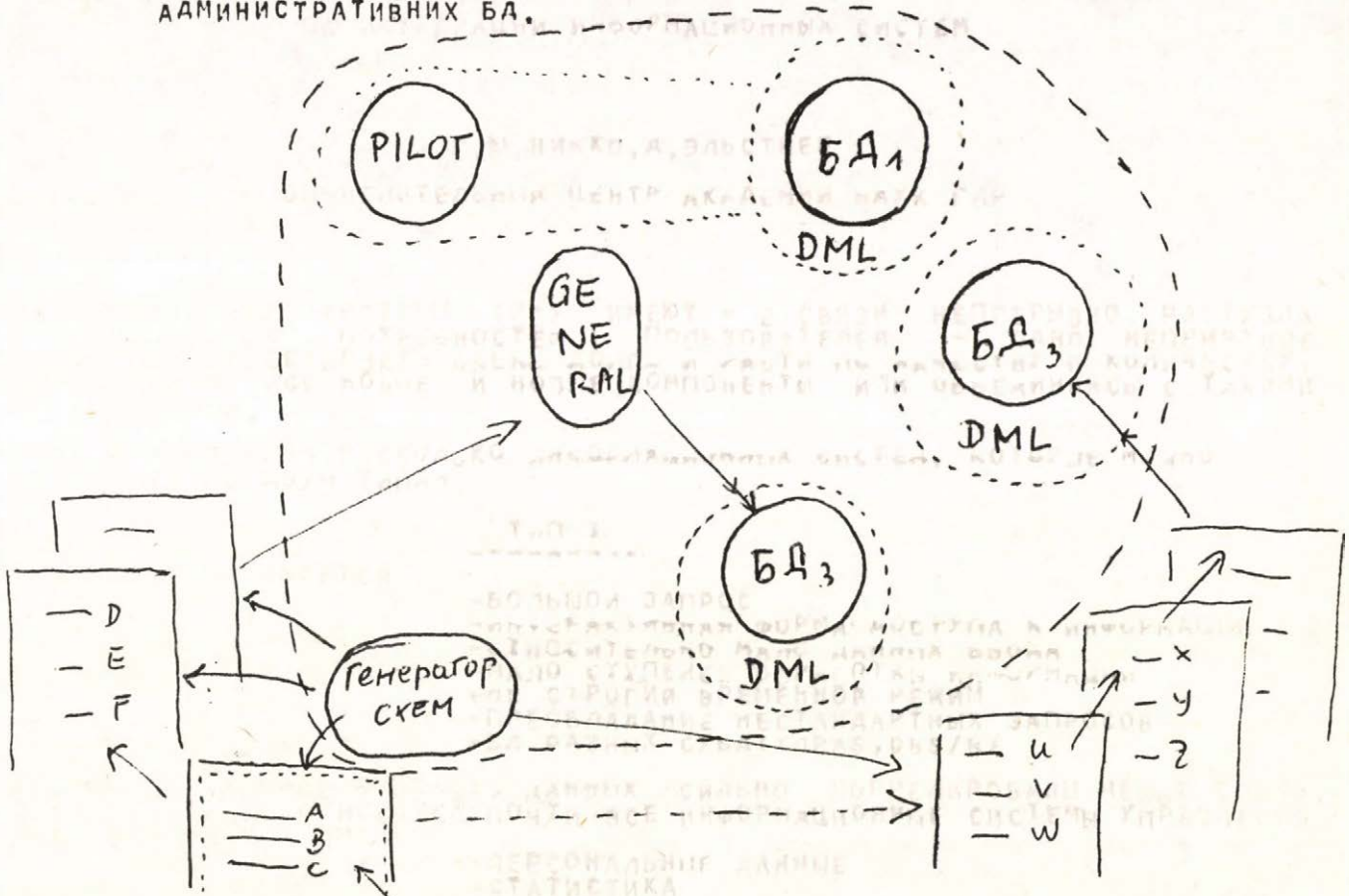
ПЕРВЫЙ ПУТЬ БЫЛ ВЫБРАН В ХОДЕ РЕАЛИЗАЦИИ -АУП-. НО КОМФОРТАБЕЛЬНОСТЬ УПРАВЛЕНЧЕСКОЙ ПРОГРАММЫ ОПЛАЧИВАЕТСЯ БОЛЬШИМИ ЗАТРАТАМИ СИЛ ПРИ РАЗРАБОТКЕ И ПРИ РОГРАММИРОВАНИИ В СЛУЧАЕ ИЗМЕНЕНИЙ ПРОЦЕДУРЫ ОБРАБОТКИ ИНФОРМАЦИИ.
НАЧИНАЯ С ВЕРСИИ OS 7.1 ИМЕЕТСЯ ВОЗМОЖНОСТЬ СОЗДАВАТЬ ТАК НАЗЫВАЕМЫЕ "СЕТИ ЗАДАЧ".
ТАКИМ ОБРАЗОМ СОЗДАНИЕ "КОНТРОЛЬНОЙ ПРОГРАММЫ" ТРАНСФОРМИРУЕТСЯ В СОЗДАНИЕ СООТВЕТСВУЮЩЕЙ СЕТИ И ВКЛЮЧЕНИЕ СПЕЦИАЛЬНЫХ ТИПОВЫХ ПРОГРАММ ДЛЯ ВЫЗОВА СООТВЕТСТВУЮЩИХ СЛУЖБ -АУП-.

КАРТИНА 3: ПРИМЕР ОБРАБОТКИ "СЕТИ ЗАДАЧ" ПОД -АУП-.



ОТНОСИТЕЛЬНО ПРОБЛЕМ СИСТЕМ ТИПА II - ПЕРВЫЕ ИДЕИ РЕАЛИЗАЦИИ ТОЛЬКО ЧТО СФОРМУЛИРОВАЛИСЬ. ЦЕЛЬ СОСТОИТЬ В ТОМ, ЧТОБЫ СОЗДАТЬ ЕДИНЫЙ ПОДХОД В РЕАЛИЗАЦИИ ДИАЛОГОВОЙ РАБОТЫ С СИСТЕМАМИ БАНКА ДАННЫХ DBS/R И TOPAS ,ТАК НАЗЫВАЕМУЮ СИСТЕМУ ПОЛЬЗОВАТЕЛЯ. ПЛАНИРУЕТСЯ ЕДИННЫЙ ВЗГЛЯД ПОЛЬЗОВАТЕЛЯ ДЛЯ ВСЕХ АДМИНИСТРТИВНЫХ БД, ЧТОБЫ ОБЛЕГЧИТЬ РАБОТУ С НИМИ И ТЕМ САМИМ ПОВЫСИТЬ ИХ ПРИЕМЛЕМОСТЬ. НЕОБХОДИМОСТЬ ТАКОЙ СИСТЕМЫ ВОЗНИКАЕТ ИЗ ЗА ТОГО, ЧТО ДИАЛОГОВОЙ ДОСТУП ПОЛЬЗОВАТЕЛЯ К ИНФОРМАЦИИ И ИЗМЕНЕНИЕ СОДЕРЖИМОГО БАЗ ДАННЫХ У СИСТЕМ TOAS И DBS/R ПРЕДПОЛАГАЕТ ДОСКОНАЛЬНЫЕ ЗНАНИЯ СИСТЕМ. ТАКИМ ОБРАЗОМ СПОНТАННЫХ ЗАПРОСОВ НЕ ВОЗМОЖНО.ИМЕЮЩИЕСЯ ЯЗЫКА МАНИПУЛЯЦИИ ИНФОРМАЦИИ СИСТЕМ TOPAS И DBS/R А ТАКЖЕ ЯЗЫКА PILOT И GENERAL ТРЕБУЮТ СПЕЦИАЛЬНЫХ ЗНАНИЙ, КОТОРЫХ НЕТ У КОНЕЧНОГО ПОЛЬЗОВАТЕЛЯ. ИСХОДНЫМ ПУНКТОМ ПРИ СФОРМУЛИРОВАНИИ ОСНОВНЫХ ИДЕЙ БЫЛО ПРИМЕНЕНИЕ ЭТОЙ СИСТЕМЫ НАИВНЫМ ПОЛЬЗОВАТЕЛЕМ, НО ПРЕДПОЛАГАЮТСЯ И СПЕЦИАЛЬНЫЕ ВОЗМОЖНОСТИ ДОСТУПА ДЛЯ АДМИНИСТРАТОРА БД И ПРОГРАММИСТА. РАЗРАБАТИВАЕТСЯ МОНИТОР, КОТОРЫЙ УПРАВЛЯЕТ ДИАЛОГОМ, СОЗДАЕТСЯ ДЛЯ ВСЕХ АДМИНИСТРАТИВНЫХ БД ЕДИНЫЙ ВЗГЛЯД ПОЛЬЗОВАТЕЛЯ. С ПОМОЩЮ СПЕЦИАЛЬНОГО КОМАНДА КАЖДЫЙ ПОЛЬЗОВАТЕЛЬ ПОЛУЧИТ ДОСТУП К СВОЕМУ БД. ГЕНЕРИРУЕМЫЕ ДИАЛОГОВЫЕ СХЕМЫ ВЕДУТ ПОЛЬЗОВАТЕЛЯ К ЗАТРЕБОВАННОЙ ФУНКЦИИ.ДЛЯ НАИЛУЧШЕЙ ПРИЕМЛЕМОСТИ ПРЕДУСМОТРЕНЫ ЕДИННЫЕ КОДЫ ВОЗВРАЩЕНИЯ,ОШИБОК И ДРУГИХ РЕАКЦИИ ПРОГРАММ.

КАРТИНА 4:
ОСУЩЕСТВЛЕНИЕ ЕДИННОГО ВЗГЛЯДА ПОЛЬЗОВАТЕЛЯ ДЛЯ СУЩЕСТВУЮЩИХ АДМИНИСТРАТИВНИХ БД.



ЗАЩИТА ИНФОРМАЦИИ БУДЕТ ДВУХСТУПЕНЧАТОЙ, ВО ПЕРВЫХ ПРИ НАЧАЛЕ СЕАНСА СПЕЦИАЛЬНЫМ КОДОМ ПОЛЬЗОВАТЕЛЯ И ВО ВТОРЫХ ПРИ АВТОМАТИЧЕСКИМ РАСПРЕДЕЛЕНИЕМ ПРОЕКТНЫХ ФАЙЛОВ.
СИСТЕМА БУДЕТ ИМЕТЬ МОДУЛЯРНУЮ СТРУКТУРУ, ЧТОБЫ ПОЛЬЗОВАТЬСЯ УЖЕ РАЗРАБОТАННЫМИ ЧАСТЯМИ.
НАХОЖДЕНИЕ ИНФОРМАЦИИ : ИМЕЕТСЯ СТАНДАРТНЫЕ И НЕСТАНДАРТНЫЕ ЗАПРОСЫ ДОСТУПА К ИНФОРМАЦИИ. ДЛЯ СТАНДАРТНЫХ ЗАПРОСОВ НУЖНО ПРОСТО УТОЧНИТЬ ПАРАМЕТРЫ. РЕЗУЛЬТАТ ПОИСКА ВЫДАЕТСЯ НА ЭКРАНЕ. ПРИНЦИПИАЛЬНЫХ ПРОБЛЕМ ЗДЕСЬ НЕ ВОЗНИКАЮТ.

УДОВЛЕТВОРЕНИЕ БОЛЬШОГО КРУГА НЕСТАНДАРТНЫХ ЗАПРОСОВ ДОСТИГАЕТСЯ ТЕМ, ЧТО ОТ ПОЛЬЗОВАТЕЛЯ СФОРМУЛИРУЕТСЯ ПРИ СОЗДАНИИ ЭТОЙ СИСТЕМЫ МНОЖЕСТВО ТАК НАЗЫВАЕМЫХ "ТИПОВ ЗАПРОСОВ", КОТОРЫЕ РЕАЛИЗУЮТ ОДИН ВЗГЛЯД ПОЛЬЗОВАТЕЛЯ.
КАЖДОМУ ТИПУ ЗАПРОСОВ СООТВЕТСТВУЕТ ОДНА ФУНКЦИЯ В СИСТЕМЕ ПОЛЬЗОВАТЕЛЯ.
УТОЧНИВ СПЕЦИАЛЬНЫЕ ПАРАМЕТРЫ, ПОЛЬЗОВАТЕЛЬ МОЖЕТ ПРИ ВЫЗОВЕ ФУНКЦИИ ВЫБРАТЬ АКТУАЛЬНЫЙ ВЗГЛЯД ПОЛЬЗОВАТЕЛЯ, ТАКИМ ОБРАЗОМ КОМБИНИРУЯ ЭТИ ПАРАМЕТРЫ МОЖНО РЕАЛИЗОВАТЬ БОЛЬШОЕ КОЛИЧЕСТВУ ЗАПРОСОВ. ЕСЛИ НЕТ СООТВЕТСТВУЮЩЕЙ ФУНКЦИИ, ТО ПРЕДОСТАВЛЯЕТСЯ ВОЗМОЖНОСТЬ МОДИФИЦИРОВАТЬ СУЩЕСТВУЮЩИЕ ФУНКЦИИ.

БУДУТ СОЗДАНЫ СПЕЦИАЛЬНЫЕ МЕХАНИЗМА, КОТОРЫЕ ГАРАНТИРУЮТ ФУНКЦИИ "UPDATE" ДЛЯ РАЗЛИЧНЫХ ДАННЫХ.

MTA SZTAKI Tanulmányok 203/1988 p.83-96

ЛОГИСЧЕСКИЕ ПРОБЛЕМЫ ИНФОРМАЦИОННЫХ СИСТЕМ

Ярослав Покорны

Центральный вычислительный
институт ЧСАН

Под водаренскоу Вежи 2
Прага
Чехословакия

Введение

Существует тезис об этом, что хуже развала системы является некорректное поведение системы. В случае информационных систем можно понимать некорректность как случай, в которых пользователь получает неправильные ответы на свои запросы.

Многообразие некорректности нельзя описать формальным способом без формализации информационной системы. Так как корректность связана с логическими понятиями, надо прежде всего описать информационную систему средствами языка логики.

Вообще говоря, логика является полезна для каждой информационной системы. Опираясь на язык логики первого порядка, мы попытаемся анализировать точнее проблемы, связанные с корректностью информационных систем. Уже применение этой логики пред-

ставляет компромисс при исследовании корректности. Конечно, разработка средств логики первого порядка позволяет получить непосредственно много результатов, которые в связи с описанием информационной системы могут помочь понять проблемы корректности.

Немало проблем связано с логической сутью информационной системы. Среди них можно назвать:

1. настоящий мир более сложнен, чем его модель (информационная система);
2. информационная система некорректна с логической точки зрения;
3. логические свойства информационной системы порождают сложные алгоритмические проблемы;
4. применение информационной системы связано с концептуальными проблемами (например, трансформация предложений на естественном языке в предложения искуственного языка или интерпретация запросов).

Мы будем заниматься в этой работе первыми тремя классами проблем, так как они взаимно связаны. Например, когда ограничимся на классы структур, ведущих к полиномиальным алгоритмам обработки, ясно, что упрощаем отношения, существующие в предметной области, модель который мы конструируем.

Целью работы является показать некоторые теоретические ограничения информационных систем. На практике это значит, что например проектирование концептуальной схемы базы данных, выбор конструктов языков запросов или выбор аксиом в дедуктивных подходах (см. базы знаний, экспертные системы) надо приспособить этим ограничениям.

В гл. 1 описана совокупность понятий, связанных с логикой первого порядка. В гл. 2 обсуждаются логические подходы к реляционной модели данных, поскольку именно эта модель явно связана с понятиями логики первого порядка. В гл. 3 выводим некоторые результаты об алгоритмической сложности возможных информационных систем.

В гл. 4 рассматривается понятия корректности (выходящие из работ Лундберга - см. например /1/). В заключении сформулированы основные проблемы и результаты, которые являются важными для конструкции информационных систем новых поколений.

1. Основные понятия логики

Опишем язык логики первого порядка $(\Sigma, \emptyset)$, где Σ - алфавит, включающий символы: (,) (скобки); $x, y, w, x_1, \ldots$ (индивидуальные переменные); $a, b, \ldots$, (константы); $\neg, \vee, \wedge, \rightarrow, \forall, \exists$ (связки и кванторы); $R_1, R_2, \ldots$, (предикатные символы) и $\emptyset$ - множество правильно построенных формул (в дальнейшем только формул), определенных следующим способом:

/i/ $R_i(S_1, \ldots, S_n) \in \emptyset$

где R_i - n-арный предикатный символ
и S_j - переменная или константа;

/ii/ если w и w_1 - формулы, то $\neg w$, $w \wedge w_1$
$w \vee w_1$, $w \rightarrow w_1$, $\forall x\,(w)$, $\exists x(w)$ тоже формулы

/iii/ других формул нет.

Позивитным литералом называется формула типа /i/, т.е. $R_i(S_1, \ldots, S_n)$, негативный литерал имеет форму $\neg R_i(S_1, \ldots, S_n)$. Замкнутые и открытые формулы определяются обыкновенным способом, т.е. открытые формулы содержат по крайней мере одну переменную, несвязанную кванторами $\forall, \exists$ (такие переменные называются свободными в w).

При обыкновенной интерпретации связок и кванторов, заменой R_i отношениями и констант элементами кокого-то множества индивидуумов D (здесь это будут индивидуальные константы, представленные в базе данных), замкнутые формулы могут быть истинными или ложными. Замкнутые формулы называются высказаниями. Высказания получим тоже из открытых формул w после оценки свободных переменных в w индивидуумами.

Интерпретация языка (, Ø), в которой такая-то формула w выполнена (т.е. она справедлива для всех возможных оценок свободных переменных), называется моделью w .

Язык (Σ , Ø) вместе с логическими акспомами и двумя правилами вывода (modus pone s и обобщение) составляет исчисление предикатов первого порядка. Включением дальнейших аксиом (так называемых нелогических аксиом) T ⊂ Ø получим теорию первого порядка T. Если w ∈ Ø выполнена в каждой модели T, говорим, что w является логическим следствием T (обозначаем T ⊨ w). Формула w , выводима из T правилами вывода (обозначаем T⊢ w), если она возможно порождается конечным набором применений правил вывода. Известным результатом является факт, что для теорий первого порядка справедливо T⊨ w тогда и только иногда, когда T⊢ w . Проблема только в том, что в общем случае не существует алгоритм, который в случае когда T⊬ w , дает ответ в конечном времени.

Обратим теперь внимание на реляционные базы данных, где логика предикатов и теории первого порядка играют значительную роль однако с некоторыми ограничениями общего случая.

2. Логический подход к реляционной модели данных

Реляционный язык L_R возникнет из общего языка (Σ , Ø) в результате следующих ограничений:

количество констант конечно
количество символов предикатов конечно
введены типы (как унарные предикаты)
кванторы ограничены
один из символов предикатов "=" (равенство)

Интерпретация L_R тоже ограничена. От любого I языка L_R требуе , чтобы каждому индивидууму из D соответствовала только одна константа из L_R и = (a, a), что справедливо для каждого a ∈ D (см. например /2/).

Отметим, что L_R также называют доменовое реляционное

исчисление (см. например /3/).

Реляционная схема $\mathcal{R}$ определяется как пара (L_R, IO) где IO совокупность замкнутых формул из L_R , называющихся ограничения целостности. Любая интерпретация L_R назявается реляционная база данных. В соответствии с $\mathcal{R}$ обозначаем ее символом $\mathcal{R}^*$. В зависимости от IO мы интересуемся только допустимыми $\mathcal{R}^*$, в которых $\mathcal{R}^*$ является моделью IO. Отметим, что формулы из IO представляют "фильтры", посредством которых получим данные, соответствующие существам и их взаимосвязям в предметной области.

Запрос на $\mathcal{R}$ определяется выражением

$$\{x_1,\ldots, x_n \mid A(x_1,\ldots,x_n)\} \qquad /*/$$

где $A \in L_R$, X_i - переменные, свободные в A. Ответом на запрос /*/ называем n-арное отношение, картежи которого, используемые как оценки переменных $x_1,\ldots x_n$, определяют значение T (истина) для A.

Этот подход к запросам на $\mathcal{R}$ основан на модели $\mathcal{R}$, т.е. он связан со семантикой $\mathcal{R}$. Иногда называют его семантическим.

В / 2/ показана вторая возможность - так называемый синтактический подход. Так как для $R^* \in \mathcal{R}$ выражение $(a_1,\ldots,a_n) \in R^*$ можно в L_R (или в $(\Sigma, \emptyset)$) заменить литералом $R(a_1,\ldots a_n)$, конструируем теорию T_R , которая включает все также позитивные литералы, получаемые из $\mathcal{R}^*$, аксиомы сровнения и несколько аксиом, которые будут объяснены в Гл. 4. Формулы из IO опять функционируют как фильтр. Реляционная теория T_R позволяет выводить все те формули IO, которые были выполнены в $\mathcal{R}^*$ (см. определение модели w), и, обратно, если $T_R \vdash w$, потом w выполнена в $\mathcal{R}^*$. Конечно, T_R имеет только одну модель, которая совпадает с $\mathcal{R}^*$. Если $w \in IO$, потом w справедлива в $\mathcal{R}^*$ тогда и только тогда, когда $T_R \vdash w$.

Очевидно, что с точки врения коммуникации мы не получим сильнее средства, чем известные реляционные языки запросов (см. например /3/). Это значит, что селективная сила таких средств не меньше, чем у языка реляционной алгебры. Припомним,

что запросы этих (реляционно полных) языков требуют для вычисления полиноминальное время. Однако не все полиномиальные запросы можно выразить в этих языках (напр. операция транзитивного замыкания бинарного отношения).

Прежде тем, чем заняться развитием синтактического подхода, ответим на вопрос, какие языки запросов, основанные на L_R обладают свойством, что они и только они выражают все полиноминальные запросы. Иммерман в /4/ определяет такой язык. Он является расширением L_R на конструкт наименьшей неподвижной точки(least fixpoint) вместе с предположением, что на D определен полный порядок " $\leqq$ ".

Отметим, что в этих подходах нельзя работать с дизъюнктивной информацией (например $R(a) \vee S(a)$), проблемы возникают и в случае существования так называемых нулевых значений в картежах отношений.

Чтобы повысить выразительную силу данного формализма, надо смотреть на базу данных не только как на теорию базисных фактов (позитивные литералы), а на теорию, где T содержает дальнейшие аксиомы. Этот подход известен под названием дедуктивный. Сюда входят формулы в специальной, так называемой клаузуларной нормальной форме.

Формула клаузуларной н.ф. имеет вид:

$$\forall x_{i1} \cdots \forall x_{is} \; (w_1 \wedge \ldots \wedge w_k \rightarrow v_1 \vee \ldots \vee v_p)$$

где w_i, v_j - позитивные литералы. Отметим, что существует алгоритм, который каждую формулу $w \in L_R$ отображает на $w' \in L_R$, которая в клаузуларной н.ф. Известно, что эта трансформация не сохраняет логическую эквивалентность, а только восполнимость (т.е. w и w' современно выполнимы или нет).

Так как кванторы существования были исключены уже в алгоритме трансформации, можно записать w' в виде

$$w_1 \wedge \ldots \wedge w_k \rightarrow v_1 \vee \ldots \vee v_p$$

или эквивалентным способом как

$$\neg w_1 \vee \ldots \vee \neg w_k \vee v_1 \vee \ldots \vee v_p$$

Из множества разных случаев можно выделить некоторые специальные виды формул, которые играют значительную роль при конструкции дедуктивных баз данных или баз знаний.

/i/ $k = 0,\ p = 1$ элементарные факты
Пр.: работает-на-задачи (Иван, B20)

/ii/ $k = 1,\ p = 0$ негативные элементарные факты
Пр.: $\neg$работает-на-задачи (Иван, B20)

/iii/ $k > 1,\ p = 0$ IO
Пр.: $\neg$(мужчина (x) $\wedge$ женщина (x))

/IV/ $k > 1,\ p = 1$ IO или аксима (правило вывода)
Пр.: $\neg$(работает-на-задаче (x,y) $\wedge$ работает-на-задаче (x,z) $\longrightarrow$ y = z)

Замечание: эквивалентное выражение формулы :
работает-на-задаче (x ,y) $\wedge$ работает-на-задаче (x ,z) $\longrightarrow$ y = z
известно под названием функциональная зависимость.

/V/ $k = 0,\ p > 1$ дизъюнктивная информация
Пр.: работает-на-задаче (Иван, B20) $\vee$ работает-на-задаче (Иван, C30)
(не знаем на которой)

/VI/ $k \geqslant 1,\ p > 1$ IO или аксиома (правило вывода)
Пр.: зарплата (Иван, y) $\wedge$ y > 500 $\longrightarrow$
работает-на-задаче (Иван, B60) $\vee$
работает-на-задаче (Иван, B70)

/VII/ праздная клаузула $\square$

При конструкции дедуктивной базы данных надо соблюдать особую осторожность в случаях /IV/ и /VI /, т.е. рассматривать формулу как IO или аксиому. С помощью аксиомы надо выводить новые факты. Этого нельзя, например, сказать о функциональных зависимостях (они включаются в IO).

Хорновской клаузулой (Horn clause) называется случай вида $k \geqslant 1$, $0 \leq p \leq 1$. В соответствии с формулами этого специального вида можно применять механизм резолюции, т.е. метод вывода, обобщающий классические правила вывода - modus ponens и обобщение.

Принцип метода резолюции прост / 5/ :
Пусть заданы две клаузулы

$$p_1 \vee \ldots \vee p_i \vee q \quad , \quad r_1 \vee \ldots \vee r_j \vee \neg q \qquad /\text{жж}/$$

Если существует замена переменных в q и в $\neg q$ переменными или константами, ведущая к унификации литералов q и $\neg q$, то формула

$$\tilde{p}_1 \vee \ldots \vee \tilde{p}_i \vee \tilde{r}_1 \vee \ldots \vee \tilde{r}_j,$$

где $\tilde{p}_i, \tilde{r}_j$ обозначают литералы p_i, r_j, после замены, справедлива тогда и только тогда, когда справедливы клаузулы /жж/.

Применение резолюции к установлении вывода $T \vdash w$ делается на множестве формул $\{T, \neg w\}$. Если после конечного числа применений правила резолюции закончим на $\square$, то $T \vdash w$ Метод резолюции полный в том смысле, что справедлива тоже импликация - если $T \vdash w$, потом алгоритм метода заканчиваются после конечного числа шагов на праздной клаузуле. В остальных случаях нельзя ничего сказать о выводимости w.

Вообще резолюция на нехорновских теориях ведет к очень неэффективным алгоритмам. Хорновское ограничение позволяет конструктировать экспоненциональные алгоритмы.

Отметим, что язык PROLOG в настоящем времени представляет из себя сильное средство использующее указанные идеи. В большинстве случаев внедрение этого языка не позволяют работать с большими массивами данных, хранимых на внешних ЗУ. Так как в теории без данных уже существуют эффективные алгоритмы вычисления запросов, а резолюция экспоненциальная, из этого факта вытекает следующая стратегия для обработки запросов в интегрированной системе (база данных + база знаний) или дедуктивных системах:

- в первой фазе использовать обыкновенные средства баз данных (семантический подход)

- во второй фазе использовать дедуктивные механизмы (синтактический подход).

3. Проблемы сложности алгоритмов

В теории баз данных очень бурное развитие получили так называемые зависимости- специальный случай I O. Одной из основных задач теории зависимостей является задача о принадлежности (membership problem), состоящая в установлении того, принадлежит ли логическое условие w (зависимость) замыканию Δ^+, где Δ - данное множество зависимостей. Так как большинство классов зависимостей можно выразить с помощью формул первого порядка / 6/ , требуется узнать, является ли w следствием Δ . В некоторых случаях могут быть зависимости использованы не только как I O, но как законы дедукции, они могут внести в базу данных (знаний) новую информацию.

Таким образом, для исследования алгоритмической сложности дедукции можно применить некоторые результаты из теории зависимостей.

Рассмотрим эти зависимости. Зависимостью называется формула вида

$$\forall y_1 \ldots y_k \; \exists x_1 \ldots x_e (A_1 \wedge \ldots \wedge A_p \longrightarrow B_1 \wedge \ldots \wedge B_q)$$

где: /1/ $k, p, q \geqslant 1$, $e \geqslant 0$

/2/ A_i позитивные литералы /кроме "="/, содержащие точно $y_1, \ldots, y_k$

/3/ B_j содержат все $x_1, \ldots x_e$ и (возможно) $y_1 \ldots, y_k$

/4/ или все B_j нет "=", или $e = 0$ и все B_j литералы с "="

Условие /4/ выделяет два основных класса зависимостей - порождающие строки (TGD) и порождающие равенства (EGD). В случае $e = 0$ зависимость называется полной (FID), в обратном случае включенной.

Отметим, что функциональные зависимости (FD) принадлежат к EGD , многозначные зависимости (MVD)- к TGD . Если

$e > 0$, $q = 1$ и B_1 нет "=", то говорим о шаблонных зависимостях (template). Класс шаблонных зависимостей обозначаем ETD Важный частный случай ETD - это зависимости по включению (inclusion), которые создают класс IND . С помощью этих зависимостей можно моделировать известные IS A - отношения.

Все рассматриваемые зависимости можно свести к типованным (обозначаем TETD, TFID, ..., и т.п.).

Первый отрицательный результат касается разрешимости проблемы, в случае, если формула первого порядка эквивалентна с какой-то зависимостью. Пусть для множества X формул E (X) обозначает множество всех эквивалентных формул. Потом E (FID) и E(TGD) не являются рекурсивными / 7/ .

Дальнейшие результаты касаются проблемы о принадлежности (п.п.)

/ i / для TETD п.п. неразрешима / 8/

/ ii / для TFID п.п. можно решить во времени O ($C^{n/\log n}$) но не в O ($d^{n/\log n}$) , где $c,d > 1$ постоянные /9/

/iii/ для FID п.п. можно решить во времени O (C^n), но не в O (d^n) , где $c > d > 1$ постоянные / 9/

Следует отметить, что если считать простые экспоненциальные алгоритмы самыми сложными для применения на ЭВМ, то FID являются самым большим классом зависимостей, который можно употребить в дедукции.

Невсегда алгоритмы экспоненциальны. Многочисленные исследования посвящены выделению специальных классов зависимостей, для которых алгоритмы п.п. полиноминальны (например FD и MVD).

Отношения сложности в зависимостях очень интересны. Например, для IND п.п разрешимая, но для расширения IND $\cup$ FD уже не разрешимая.

4. Проблемы корректности информационной системы

Чтобы получить корректный ответ на запрос, то надо предполагать, что и система корректна. Оказывается, что в большинстве случаев нет средств для доказательства корректности какой-то информационной системы.

В /1/ под корректностью понимается выполнение трех условий:

/i/ согласование системы
/ii/ выполнимость в настоящем мире объектов
/iii/ полнота системы

Прежде всего мы опишем информационную систему множеством высказаний. В базах данных выделяем обыкновенно подмножество этих высказаний - концептуальную схему (К-схему).

Множество высказаний (или теория) T <u>согласовано</u>, если одновременно недействительно $T \vdash w \wedge T \vdash \neg w$ для любой формулы w. Для теорий первого порядка это эквивалентно предложению, что T имеет модель. Отметим, что в реляционной модели данных эта модель представлена $\mathcal{R}^*$. Подчеркнем, что эта модель удовлетворяет только I O ("фильтрование" данных), которые могут недостаточно полно описать настоящий мир объектов. Из этого вытекает, что поскольку выполнено условие /i /, то выполнение условия /ii/ не всегда можно обеспечить. Далее, T является <u>полной</u>, если всегда $T \vdash w$ или $T \vdash \neg w$, для любого w. Отметим, что теории в большинстве случаев неполные.

Улучшение положения в этой области способствовало введением Рейтером /2/ трех типов аксиом в синтактический подход:

/i/ аксиома однозначности ссылок (индивидуумы с разными именами разные)

/ii/ предположение замкнутого мира (CWA)
$(T \vdash \neg R(a_1,\ldots a_n) \leftrightarrow T \not\vdash R(a_1,\ldots a_n))$

/iii/ замкнутые домены (не существует других объектов, кроме тех, которые находятся в базе данных)

Важным случаем среди дедуктивных баз данных являются так называемые дефинитные дедуктивные базы данных. Они представляют собой теории T, включающие:

- элементарные факты (позитивные литералы)
- аксиомы из синтактического подхода
- правила дедукции вида $A_1 \wedge \ldots \wedge A_p \longrightarrow B;\ p \geqslant 0$
  (т.е. Хорновские клаузулы с единственным литералом на правой стороне)
- IO

Такие базы данных всегда согласованы под CWA. Отметим, что в случае недефинитных баз данных, т.е. таких, которые содержают дизъюнктивную информацию, это высказани несправедливо.

При конструкции К-схемы обыкновенно только предполагается согласование отвечающих формул. В некоторых применениях дедуктивных баз данных надо этот вопрос решить серъезнее, как в случае множества формул IO, так и множества аксиом (правил вывода).

К согласованию IO можно прийти с помощью следующей процедуры:
Пусть дано множество согласованных IO и хотим прибавить новую формулу w.

1. Методом резолюции пытаемся вывести $IO \vdash w$ и $IO \vdash \neg w$.
   В наилучшем случае это справедливо (и мы не добавим w к IO).
   Во всех остальных случаях нельзя сказать согласованы ли $IO \cup w$ и мы обыкновенно добавляем w к IO.
2. После этого мы хотим прибавить $\bar{w}$ и покажем на шаге 1, что $IO \cup \bar{w}$ несогласованы. Из этого не вытекает неурочность $\bar{w}$, но например существует ошибка при добавлении w. Следовательно, надо предложить тесты на согласование подмножеств данного множества IO.

   Конечно, мы знаем вторую возможность теста на согласование - существование модели (т.е. $\mathbb{R}^*$). В этом случае получим позитивный ответ, когда мы например, убеждены о правильности данных. Надо отметить, что конечно негарантирована минималь-

ность множества IO. Если у нас нет данных и мы сделаем конструкцию модели, тоже не всегда достигнем успеха. Она не должна существовать или она бесконечная.

Эти проблемы отражают факт, что существуют классы формул, у которых проблема согласования неразрешима. Пример таких формул представляет класс

$$\forall\exists\{A\}^*_w$$

где w содержает только один бинарный предикат и $\{A\}^*$ обозначает n, $n \geqslant 0$, появлений A.

Заключение

Продемонстрированные проблемы намечают большое количество трудностей, появляющихся при создании "интеллектуальных" информационных систем. Трудности возникают уже в гарантии согласования К-схемы или множества аксиом дедуктивной базы данных (знаний). Успеха можно достигнуть только в простых случаях (например дефинитные базы данных). Вторым ограничением является можность алгоритмов, которая опять ведет к только простым классам формул.

Резумируя все сказанное, можно констатировать, что точные машинные выводы (ответы) в сложной предметной области иллюзорны подобно как человеческие в настоящем мире.

Л И Т Е Р А Т У Р А

/1/ Lundberg, B.: On correctness of information systems. Information Systems, 8, 2, 1983.

/2/ Reiter, R.: Towards a logical reconstruction of relational database theory. In: Brodie, Mylopoulos, and Schmidt, Ed., On Conceptual Modelling. Springer-Verlag, 1984.

/3/ Ullman, J.D.: Principles of database systems. Sec.Ed., Computer Sc. Press, Inc., 1982.

/4/ Immerman, N.: Relational Queries Computable in Polynomial Time. Proc. 14th ACM Symp. Theory of Computing. San Francisco, 1982.

/5/ Gray, P.: Logic, Algebra and Databases. Ellis Horwood Lim. Publ., 1985.

/6/ Fagin, R.: Horn Clauses and Database Dependencies. JACM,82

/7/ Makowsky, J.A.: Characterizing Data Base Dependencies. Proc. of the 8th ICALP '81, LNCS 115, 1981.

/8/ Gurevich, Y., Lewis, H.R.: The inference problem for template dependencies. Proc. of the 14th ann. ACM Symposium of Computing, 1982.

/9/ Makowsky, J.A.: Model theoretic issues in theoretical computer science, part 1: Relational databases and abstract data types. Logic coll. '82, Elsevier Sc. Publ. (North Holland), 1984.

MTA SZTAKI Tanulmányok 203/1988 p.97-113

СЕМАНТИЧЕСКИЕ МОДЕЛИ БАЗ ДАННЫХ

Ярослав Покорны
Центральный вычислительный
институт ЧСАН
Под водаренскоу вежи 2,
Прага
Чехословакия

Введение

Каждая система управления базой данных (СУБД) основана на какой-либо модели баз данных. С развитием теории баз данных (БД) развивалось и понятие модели данных или модели баз данных. В начале периода обработки данных, который начался вместе с развитием вычислительных машин, было можно говорить только об одной модели данных - файле. Пользователь отображает "объекты внешнего мира" в объекты, которые являются предметами обработки в вычислительных системах. Эти объекты называются данные. Их возможно группировать в записи, совокупность которых создает файл.

Естественно, что файл представляет очень примитивную модель данных. Семантика данных, хранемых в файле, целиком зависит от интерпретации пользователя. В лучшем случае, например, у горизонтально и вертикально однородных файлов, возможно интерпретировать записи файла как объекты предметной области. Однако затруднение заключается в том, что некоторые записи не представляют естественные объекты или взаимосвязи этих объектов. Системы, основанные только на файлах (файловые системы), оказались неадекватными для эффективных информационных технологией.

С точки зрения абстракции надо различить логическую и физическую запись. Так как логическая запись абстрагируется от деталей внедрения, можно считать файлы логических записей первым средством моделирования внешнего мира.

В конце 60-ых годов были описаны три так называемые основные или традиционные модели баз данных: иерархическая, сетевая и реляционная. Первые две позволяли пользователю явным способом определить связи между файлами. Результатом проектирования таких связей является структура данных - схема, которая представляет из себя с теоретической точки зрения дерево или ориентированный граф.

Следует отметить, что понятия модели данных (или модели баз данных) упрочняется. Модель данных - это метаязык, позволяющий строить языки описания конкретных моделей данных /1/. Например, философия CODASYL DBTG /2/, которая поддерживает сетевую модель, является первым шагом к построению различных сетевых СУБД (IDMS , TOTAL и т.п.).

С другой стороны, реляционная модель данных представляет из себя модель иного рода. Она прямо обобщает понятие однородного файла. Связи между данными опять неявные, картеж реляции - это прямый эквивалент записи.

Отличие реляционной модели данных от первых двух фундаментальное:

- физическая и логическая независимость данных превышает прагматически ориентированные сетевые и иерархические подходы;
- прямая связь с математической логикой и алгеброй дает до сих пор невиданные средства для обработки данных (языки запросов).

Отметим, что в первых годах существования реляционной модели данных ее пропагандисты не занимались семантикой данных. Описание допустимых состояний реляционной базы данных было поддержено только спецификацией и проверкой функциональных (или многозначных) зависимостей.

Тенденция приблизить формальное состояние базы данных состоянию внешнего мира (или другими словами - построить модель внешнего мира) привела исследователей к новому понятию - ограничения (условии) целостности. Так как реляционная модель данных прямо связана с исчислением предикатов первого

порядка, оказалось просто формулировать ограничения целостности (ОЦ) на этом языке.

Подмножества ОЦ были разработаны во многих работах как зависимости (по соединению, иерархические, ключевые, шаблонные и другие). Важное свойство всех этих зависимостей заключается, что они могут быть выражены формулами языка предикатов первого порядка.

Использование реляционной модели данных вместе с языками для описания ОЦ дает могучее средство для описания внешнего мира с помощью схемы реляционной базы данных (совокупность схем реляций и ОЦ) и всех картежей реляций. Возникает естественный вопрос, достаточно ли это для конструкций эффективной и удобной для пользователя системы. Оказалось, что языки с селективной силой реляционной алгебры (или реляционного исчисления) теряют много из своих достоинств в случае "плохо" спроектированной схемы реляционной БД. Алгоритмы проектирования схемы (синтез и декомпозиции) приводят к построению схемы в нормальных формах, но в общем случае в результате нет схем реляций, которые естественно описывают внешний мир.

Новый этап развития моделей баз данных начинается с 1976 г. известной работой Чена /3/, в которой были синтезированы постепенно появляющиеся тенденции в работах Абриаля, Сенко, Шмида и Свенсена, которые решали разные семантические проблемы реляционной модели. Ченова модель сущностей и связей положила начало построения, развития и теоретических оснований так называемых объектно-ориентированных моделей баз данных.

Основным понятием этих моделей является объект (или сущность). Объекты и их взаимные отношения доступные наблюдению в реальном мире (с философической точки зрения мы говорим о наивном реализме /4/ или об объективистическом подходе /5/), могут быть представлены различными способами с помощью разных конструктов из репертуара средств отдельных моделей.

В течение последних десяти лет появилось множество таких моделей. Они так несут название семантические или

концептуальные. Понятие концептуальная схема было введено в предложении ANSI/X3/SPARC /6/, описающим унифицированный подход к архитектуре СУБД. Проблема оказалась в том, что было неясно понятие концептуальной схемы вообще. Концептуальная схема представляла собой один уровень абстракции между реальным миром и данными в базе данных. По /6/ она стоит между пользовательскими "точками зрения" (внешние схемы) и внутренной схемой, описывающей реализацию базы данных.

Хотя концептуальная схема, как правило, относится к языковой конструкции, трудно сказать, описываем ли мы с ее помощью реальный мир или содержание БД. Некоторые стандарты были положены в работе группы ISO /7/, где под концептуальной схемой понимается описание предметной области (в /7/ ее называют системой объектов).

В настоящей работе, когда говорим о понятии концептуальной схемы имеем в виду следующие:

- конструкции разных концептуальных моделей БД и основание их математической теории;
- нахождение связей между естественным языком и концептуализацией с помощью средств концептуальной модели;
- нахождение методов проектирования концептуальных схем и средств для определение (и верификацию) таких характеристик, которые оформляют "хорощую" схему;
- обобщение концептуальных моделей таким способом, чтобы они могли описывать динамику системы, использовали средствами дедуктивного поиска ответа на запросы.

Отметим, что без строгих теоретических средств невозможно решать проблемы, как например, информационную эквивалентность БД, семантику операций над ДБ. Методы проектирования ведут к концептуальной схеме, которая корректным способом описывает предметную область (несмотря на то, что корректность нельзя вообще доказать алгорифмическими средствами /8 /). Обобщения современных моделей ведут к представлению знаний. Сюда принадлежат напр. и такие требования как обработка

неполной информации и моделирование времени.

В дальнейшей части работы /Гл. 1/ мы будем заниматься семантическими моделями более подробно. Будут описаны различные конструкты этих моделей и некоторые заключительные соображения.

За последние годы в ЧССР было создано много работ, посвященных теории и методологии проектирования БД. Эти работы основаны на функциональном подходе с помощью интенсиональной логики и лямбда - исчисления /9/, /10/, /11/, /12/, /13/. Функциональный подход /Гл. 2/ завершает эту работу.

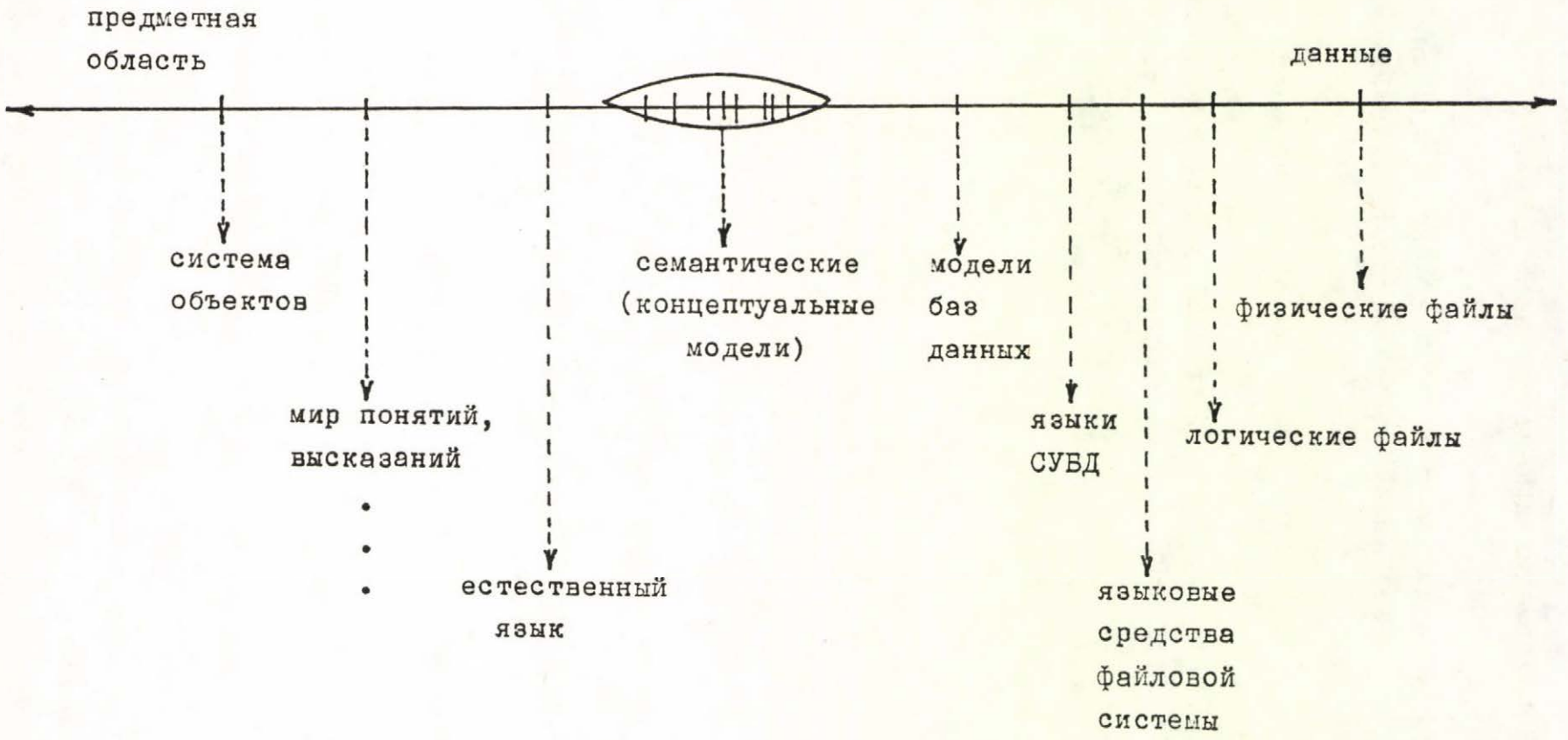
1. ПЕРЕЧЕНЬ СЕМАНТИЧЕСКИХ МОДЕЛЕЙ БД

Как указывалось в введении, под концептуальной схемой можно понимать описание части предметной области, которая называется в терминологии ISO система объектов. Это описание было бы возможно выразить, в конце концов, с помощью естественного языка, но эта идея приводит нас к таким проблемам как напр. решение проблем анализа предложений этого языка лингвистическими средствами. Очевидно, что искусственный язык концептуальной модели будет лучше. На рис. 1 изображается "концептуальная" шкала различных уровней абстракции, которые уточняют место концептуальной схемы в этой шкале.

Понятие модели данных развивалось одновременно с развитием семантических моделей. Во всех случаях мы можем всегда сказать, что модель данных задан множеством общих конструктов (например атрибут, реляция, отношение ISA и так далее) и методом их применения. С точки зрения языков модель данных представляет из себя грамматику в результате применения которой возникают схемы.

Описанный подход не говорит ничего о семантике концептуальной схемы, поскольку она является только текстом

Рис. 1


предметная область
данные
система объектов
мир понятий, высказаний
естественный язык
семантические (концептуальные модели)
модели баз данных
языки СУБД
языковые средства файловой системы
физические файлы
логические файлы

в каком-то языке. В /7/ определяют концептуальную схему как согласованную совокупность предложений, выражаемых все необходимые высказания, которые справедливы в системе объектов.

Если мы представим себе систему объектов в течении времени, когда некоторые объекты возникают и прекращают свое существование, можно ее разделить на частичные миры объектов. Необходимые высказания потом такие, которые справедливы во всех частичных мирах объектов. База информаций дана вследствие этого остальными высказаниями, справедливыми в одном данном частичном мире объектов.

Хотя эта философия является не точно определенной, она дает одну ясную перспективу: надо всегда подразумевать под концептуальной схемой какие-то высказания. Корректность совокупности таких высказаний можно потом перевести на логические проблемы, известные например из информационно-логических подходов /14/ (согласование, полнота, существование модели).

Аппарат семантических моделей дает кроме основных общих конструктов еще логический язык для формулировки некоторых важных высказаний о системе объектов. Например высказание "студент имеет не больше чем одну оценку по данному предмету и то только в таком случае, если у него есть зачет по этому предмету" нельзя выразить только двумя схемами релядий:

ЗАЧЕТ (СТУДЕНТ, ПРЕДМЕТ)
ЭКЗАМЕН (СТУДЕНТ, ПРЕДМЕТ, ОЦЕНКА)

К схемам надо прибавить еще замкнутую формулу (например в модифицированном исчислении первого порядка) ОЦ:

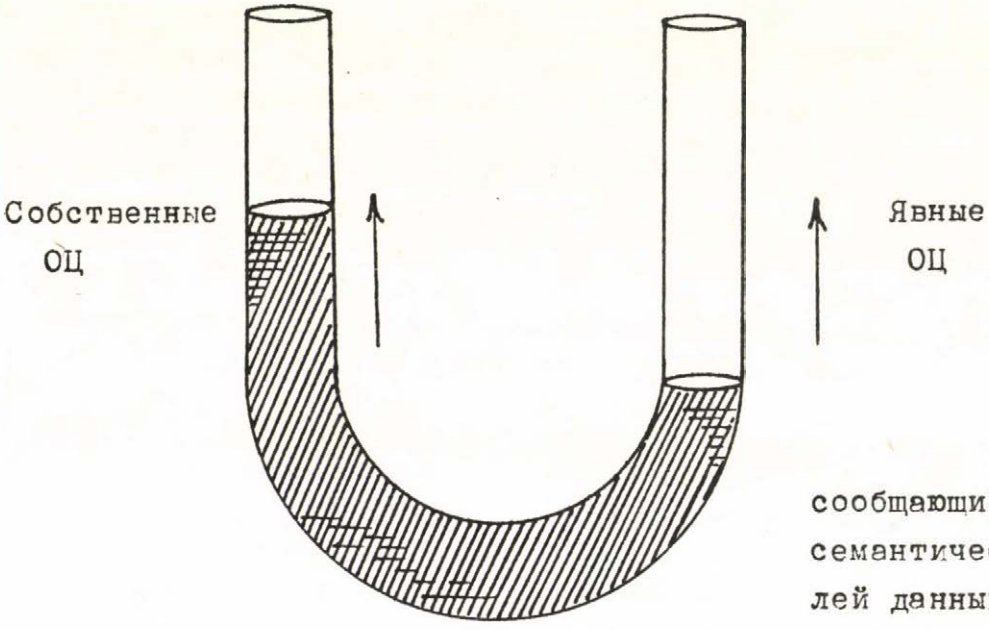
$$\forall x,y,z \quad (\text{ЭКЗАМЕН}\ (x,y,z) \longrightarrow \text{ЗАЧЕТ}\ (x,y)) \qquad /+/$$

Существование ОЦ имеет прагматический смысл. Так как концептуальная схема описывает в принципе базу информаций, на практике надо проверять вводимые данные, которые создавают базу информаций, т.е. базу данных.

В согласии с /25/ можно рассмотривать три типа ОЦ: собственные (inherent), явные (explicit) и неявные (implicit).

Примером явного ОЦ является /+/. Собственные ОЦ сосредоточены в основных конструктах, например из понятия функции вытекает, что одному элементу домены функции соотносится не более чем один элемент из множества значений. Неявные ОЦ возникают в результате применения напр. каких-нибудь правил вывода на данное множество ОЦ. Сравнение многих примеров семантических моделей данных показывает, что все объектно-ориентированные семантические модели подобны. Они отличаются друг от друга только в том, сколько ОЦ абсорбируют их основные конструкты. Чем больше ОЦ поместится в основных конструктах, тем меньше явных ОЦ надо формировать (рис. 2).

Рис. 2



сообщающиеся сосуды семантических моделей данных

Заметим, что пока мы говорим только о статических ОЦ. В некоторых современных моделях появляются также возможности использовать динамические компоненты для формулировки структурированных трансакций.

Множество собственных ОЦ в зависимости от выбора основных конструктов можно получить на основе следующего обзора некоторых известных семантических моделей:

Чен 1976 /3/: существа, связи, атрибуты
/атрибуты: простые неповторяющиеся, сложные неповторяющиеся/

Фалкенберг 1977/15/: объекты, бинарные связи, роли /объектов в связи/

Нийссен 1977 /16/: элементарные предложения

Смит & Смит 1977 /17/: иерархии обобщения
/включение IS-A иерархий в модели баз данных/

Пиротт 1977 /18/: расширения модели Чена

Чуанг 1979 /19/: /пересекающиеся множества существ, повторяющиеся простые и сложные атрибуты, и т.д./

Кодд 1979 /20/: расширенная реляционная модель (правила определенности сущности и референциональной целостности, неопределенные значения, несколько типов реляций, явные иерархии обобщения/

Хаммер, Маклеод 1980 /21/: семантическая модель данных
/абстрактные сущности, классы, связи, атрибуты разных типов, производные классы, классифицирование атрибутов, инверсия, иерархии обобщения/

Милопулос, Вонг 1980 /22/: TAXIS
/классы, метаклассы, характеристики, атрибуты, иерархии обобщения/

Шиппман 1981 /23/: функциональная модель
/множества существ, простые атрибуты - - повторяющиеся и неповторяющиеся/

Матерна, Крейчи, Покорны, Феликс, Златушка 1981 /10/: HIT
/функциональная модель основана на обобщенном понятии функции/

Хулл 1984 /24/: IFO
/объекты, группирование, декартово произведение, фрагменты (функции), иерархии обобщения/

Броди 1983-85 /25/: обобщенная методология моделей данных
/статические и динамические аспекты/

Конечно, список указанных моделей неполный, но по нему можно выделить несколько тенденций. С развиванием семантических моделей баз данных повышается множество основных конструктов в модели. Завершением этого усилия является семантическая модель Хаммера и Маклеода, применение которой уже представляет трудности для пользователя: проблема выбора конструктов (ср. с ситуацией у языков программирования).

Расширение реляционной модели данных Коддом представляет тоже завершение, которое уже сегодня не имеет продолжения.

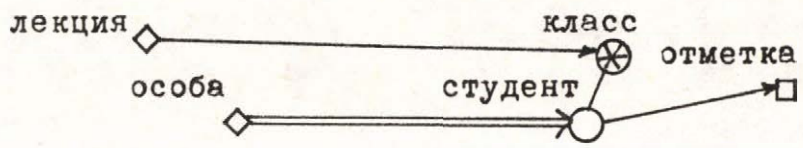
Явным является влияние результатов искусственного интелекта (Иерархии обобщения, семантические сети) и тенденции к теории /10/, /24/.

Между моделями можно различать две группы моделей:
- основанные только на понятии функции (/23/, /10/);
- основанные на разных общих конструктах типов (остальные модели).

Очевидно, что много моделей второй группы применяет также понятие атрибута (функции). Поскольку в /3/ и /18/ атрибут являлся функцией от существ, или свойств и его значениями были описательные типы, функциональные модели /23/, /10/ не учитывают это ограничение. В /24/ введена так называемая репрезентация фрагментов, с помощью которой можно выразить сложные

функциональные связи между названными структурами. Пример такой функции:

Рис. 3



где ◇ представляет абстрактный объект (существо), ○ производный объект (с помощью иерархии, обобщения или специализации), □ описательный объект, ⊛ небазисный объект, который возник группированием.
На рис. 3 лекциям присвоены классы, которые структурально являются множествами студентов вместе с функцией от студентов к отметкам.

Интересным вариантом атрибута является атрибут типа /21/, /22/ (в /24/ включен в понятии фрагмента), который множеству существ соотносит значение (например количество-студентов-в-классе).

Между нефункциональными конструкциями кроме группирования значительное место занимает агрегация, с помощью которой представляем подмножества декартова произведения множеств (структурированных) объектов.

Кроме бесспорных достоинств семантических моделей баз данных, их продолжающая разработка свидетельствует о многих проблемах и недостатках современного концептуального моделирования. Самыми важными являются:

1. неявная связь с естественным языком
2. недостаточные формальные средства
3. недостаточные средства для манипулирования концептуальными объектами
4. отсутствия методов проектирования

Из /1/ вытекают семантические проблемы интеграции баз данных (видов, точек зрения), трудное определение срезов и неестественные конструкции. Отсутствие формального аппарата /2/ в большинстве моделей ведет к определению языка концеп-

туальной схемы с помощью примеров (в случае синтаксиса, но и семантики). Отсутствие сильных средств манипулирования приводит к проблемам, связанным с трансформацией операционной семантики (концептуальная схема - /логическая/ схема базы данных). К той же группе проблем относится и трудная конструкция языков запросов над семантическими моделями.

Наконец, с /4/ связаны проблемы альтернатив в применении конструктов (семантический релятивизм). Проблемы завершает трудно отвечаемый вопрос - что такое "хорошая" концептуальная схема.

2. ФУНКЦИОНАЛЬНЫЙ ПОДХОД К КОНЦЕПТУАЛЬНОМУ МОДЕЛИРОВАНИЮ

Концептуальное моделирование какой-нибудь предметной области можно разделить на следующие шаги:

а) описание мира на естественном языке
б) анализ предметной области
в) проектирование концептуальной схемы

В /а/ и в /б/ надо работать с естественным языком выражения которого в некоторых случаях были преобразованы в предикаты и формулы исчисления предикатов первого порядка (см. напр. /14/. Отметим, что наоборот в большинстве случаев применяются прямо конструкты семантической модели, выбор которых определяется существующей методологией проектирования.

При использовании этих средств возникают некоторые проблемы:

- язык 1-ого порядка не способен выразить различия между интенсией и экстенсией, между классом и свойством. Трудно выразить объекты "высшего" типа, как например свойства высказаний;
- смысл синтактического выражения зависит от состояния внешнего мира.

Например выражение "В Варне происходит первое совещание РГ-25" понимают екстенсионалисты как предложение, которое

принимает значение TRUE или FALSE. Наоборот, в интенсиональном подходе это предложение, которое принимает значение высказания (это значит концептуальный объект) причем под высказанием понимается частичная функция от возможных миров и моментов времени в множестве $\{T, F\}$.

Интенсиональный подход к анализу естественного языка и его применению для целей моделирования использует типованное ламбда-исчисление.

Типы T попределяются следующим способом:

1. $\mathcal{B} = \{\sigma_1, \ldots, \sigma_n\}$, $n \geqslant 1$ создает базис тривиальных типов (другими словами, это совокупность сортов)

2.1 $\mathcal{B} \subset T$

2.2 Если $\eta, \xi_i \in T$, $1 \leq i \leq n$, то $(\eta\, \xi_1 \ldots \xi_n) \in T$

3. других типов, кроме построенных по правилам 1-2, нет.

Запис $(\eta\ \xi_1 \ldots\ \xi_n)$ обозначает пространство всех частичных функцией из $\xi_1 \times \ldots \times \xi_n$ до η

Для анализа естественного языка берем так называемый эпистемический базис

$E = \{o, \tau, \iota, \omega\}$ где O - множество {TRUE, FALSE}

τ - моменты времени (или вещественные числа);

ι - индивидуалы; ω - возможные миры.

Высказание "объект A имеет тип η " принято обозначать A/η или A^{η} (так "A есть η - объект")

Примеры:

1. классы η-объектов $(o\,\eta)$ - объект (отметим, что $(o\,\eta)$ есть тип характеристической функции множества, отвечающего этому классу)

2. Арифметические и логические операции:

$>/(o\,\tau\,\tau)$, $+/(\tau\,\tau\,\tau)$, $/(ooo)$, $/(oo)$, $=/(o\,\tau\,\tau)$

3. функции - агрегаты

$SUM/(\tau(o\tau))$, $COUNT/(\tau(o\iota))$

Очевидно, что отношения реляционной модели данных являются $(o\,\xi_1 \ldots \xi_n)$ - объектами.

Вышеупомянутые понятия представляют из себя типичные интенсии. С другой стороны интенсии понимаем как объекты параметризованные возможными мирами и моментами времени.

Примеры:

1. индивидуальные концепты $((\iota\tau)\omega)$ - объекты
2. свойства $((o\iota)\tau)\omega)$ - объекты
3. высказания $((o\tau)\omega)$ - объекты
4. интенсиональные отношения $(((o\,\xi_1 \ldots \xi_n)\tau)\omega)$ - объекты

Пусть $\pi = ((o\tau)\omega)$. Так например, предмет физика $(o\pi)_{\tau\omega}$ - объект, где индекс $\tau\omega$ сокращает параметризацию τ и ω . Интенсия "отметка студента по физике" является функцией, можно GR , типа $(\tau\iota(o\pi))_{\tau\omega}$; каждому студенту (ι) и данному предмету $(o\pi)$ присвоено значение - - отметка $(\mathcal{T})$. Высказание " Отметка студента Новикова по физике составляет 3" в лямбда - исчислении выражаем термом

$$\lambda w \lambda t (GR\ (\text{"Новиков"}, \text{"физика"})\ (w,t) = 3)$$

где "физика" обозначение данной лекции (типа $(o\pi)_{\tau\omega}$), = написано обыкновенным способом (подробнее /9/, /13/).

Так как типизацию объектов естественного языка можно конструировать произвольно глубоко, очень важна в применениях интенсионального подхода сортализация. Например, для предмета $((o\pi)_{\tau\omega})$ присваим тип α .

Основным понятием концептуального моделирования является атрибут, определенный как функция типа

$$(\eta\ \xi_1 \ldots \xi_n)_{\tau\omega}$$

Значение атрибута в W, T представляет функции типа $(\eta\ \xi_1 \ldots \xi_n)$ - состояние атрибута.

Отметим, что с помощью атрибута можно описать все известные конструкты семантических моделей данных. Например, атрибут типа /21/, /22/ - это $(\xi\ o(\eta))$ - объект, Ченов атрибут отношения - это $(\eta\ \xi_1 \ldots \xi_n)$ - объект, где ξ_i типы существ и η - тип значений.

Ограничение типов атрибутов позволяет определить все известные модели систематическим и унифицированным способом. Для применения в искусственном интелекте являются очень полезным общий подход, выходящий из точного анализа естественного языка.

Данный подход использован в концептуальном моделировании на основе модели баз данных HIT /10/, /12/, где типы состояний атрибутов ограничены на

$$(\eta\ \xi_1 \cdots \xi_n)$$
$$((O\ \eta_1 \cdots \eta_m)\ \xi_1 \cdots \xi_n)$$

причем η, ξ_i, η_j сорты ($O(\iota)$ - объекты).

Лямбда - исчисление представляет из себя сильное средство для конструкции языков запросов, языков ОЦ, языков для транформаций схем.

Важной составной частью модели HIT является его методология проектирования /13/.

ЛИТЕРАТУРА

/1/ Цаленко, М.Ш.: Семантические и математические модели баз данных. Итоги науки и техники. Москва 1985.

/2/ Tsichritzis, D., Lochovsky, F.: Data models. Prentice Hall, 1982.

/3/ Chen, P., P., S.: The Entity-Relationship Model: Toward a Unified View of Data. ACM TODS, 1, 1, 1976.

/4/ Meisingset, A.:Theoretical Foundations of Conceptual Modelling. Acta Universitatis Tamperensis, ser. B, vol. 22, Tampere, 1984.

/5/ Klein, H.K., Hirschheim, R.A.: A Comparative Framework of Data Modelling Paradigms and Approaches. Comp. Journ., 30, 1, 1987.

/6/ ANSI/X3/SPARC DBMS Framework: Interim Report of Study Group on Data Base Management Systems, 1977.

/7/ Griethuysen van, ed.: Concepts and Terminology of the Conceptual Schema and the Information Base. ISO/TYC 97/ CS5/N695, 1982.

/8/ Lundberg, B.: On corectness of information systems. Information systems, 8, 2, 1983.

/9/ Materna, P., Pokorný, J.: Applying simple theory of types to databases. Information systems, 6, 4, 1981.

/10/ Materna, P., Krejčí, F., Zlatuška, J., Pokorný, J., Felix, O.: HIT - databázový model. Proc. SOFSEM´81, 1981. (In Czech).

/11/ Zlatuška, J.: HIT Data Model. A Functional Approach to Data Bases. 7th Int. Sem. on DBMS, Varna, 1984.

/12/ Zlatuška, J.: HIT Data Model. Data Bases from the Functional Point of View.VLDB´85, Stockholm. 1985.

/13/ Duží, M., Krejčí, F., Materna, P., Staníček, Z.: HIT Method of the Data Base Design. Res. Rep., Brno, 1987

/14/ Gallaire, H., et al: Logic and databases: An overview and survay. Joint rep. CERT/CGE/, Univ. of Maryland, 1983.

/15/ Falkenberg, E.: Concepts for Coexistence Approach to Database Management, Int. Comp. Symp. 77, 1977.

/16/ Nijssen, G.M.: A framework for discussion in ISO/TC 97/ CS5/WG3 and comments., ISO, 1978.

/17/ Smith, J.M., Smith, D.S.P.: Database abstractions: Aggregation and Generalization. ACM TODS, 2, 2, 1977.

/18/ Pirotte, A.: The Entity-Association Model: An Information-Oriented Data Base Model: In: Proc. Int. Comp. Symp., 1977.

/19/ Chiang, T.E.: A DBMS with an E-R Conceptual Model. Proc. of the Int. Conf. on E-R approach to System Analysis and design, Los Angeles, 1979.

/20/ Codd, E., F.: Extending the Database Relational Model to Capture More Meaning. ACM TODS, 4, 4, 1979.

/21/ Hammer, M., Meleod, D.: Database decription with SDM: Semantic Database Model. ACM TODS, 6, 3, 1981.

/22/ Mylopoulos, J., et al: A Language Facility for Designing Database - Intensive Applications. ACM TODS, 5, 2, 1980.

/23/ Shipman, D.: The Functional Data Model and the Data Language DAPLEX. ACM TODS, 6, 1, 1981.

/24/ Abiteboul, S. and Hull, R.: IFO: A formal semantic database model. TR-84-304, Univ. of South Calif., 1984.

/25/ Brodie, M.L., Ridjanovic D.: On the design and specification of database transactions. In: Brodie, Mylopoulos, and Schmidt, Ed., On Conceptual Modelling. Springer-Verlag, 1984.

MTA SZTAKI Tanulmányok 203/1988 p.115-120

An Overview of the INTERBAS Data Base Management System

Dipl.-Ing.-Oek. J.Bittner
VEB Robotron-Projekt Dresden, GDR

1. Introduction

During the last 10 years the research in the field of data base management system software has been producing numerous results, which are suitable to remove the disadvantages of existing DBMS or to create facilities of information processing with new quality. These results - especially in the field of DBMS software on mainframe systems - have been realized in an insufficient manner for different reasons up today. Not before the goals of Computer Integrated Manufacturing (CIM) emerged, essential impulses were given to the requirements for further development of DBMS software and the concepts for matching these requirements.The focus of requirements is the development of a new DBMS. Therefore, a system is developed in co-operation by VEB Robotron-Projekt Dresden and the prominent Soviet Softwarehouse ZENTRPROGRAMMSISTEM Kalinin under the name INTERBAS.

2. Basic goals

The DBMS INTERBAS is designed as a system with a long-term extensible conception. Starting from the features of modern DBMS (first stage) it will be developed via an Information Base System (second stage) to a Knowledge Base System (third stage).
The first stage is based on following goals:

General-purpose DBMS including essential facilities for supporting or interfacing CIM processes

Existing DBMS offer only parts of facilities, which are needed for CIM processes. Therefore the users are forced

- either to apply several DBMS, that means, to deal with different DBMS and to couple them one another
- or to develope the absent facilities after extensive examining and selecting a suitable DBMS.

Ability of operation of DBMS on different computers under different operating systems

INTERBAS has to support all computers with a processing width up 16 bits, produced by Kombinat Robotron. Moreover it should utilize the folllowing operating systems: OS/ES, SVM, MUTOS, DCP and others. Small versions of INTERBAS should be running on computers up 216 kbytes of main storage.

Information processes of users are often running on different computers, particulary in the field of CIM applications. The unified utilization of DBMS functions is a key factor of rationalization, independent of degree of coupling.
Coupling problems between databases on different computers will be solved easier by using INTERBAS. The efficiency of coupling is higher.

Supporting different user classes with correspondending level of data independence

You have to identify different user classes:

- end users, working with query languages
- end users, working with transactions
- application programmers, developing transactions in a high level of efficiency
- developers of other software systems
- data base administrators
- operators

Variability and adaptability

Increasing concentration of specific facilities implies increasing need to form variably the components in a software system. Simultaneously, a high level of adaptability is to be obtained, in order to tailor the DBMS INTERBAS according to the concrete conditions of an individual user and computer. A minimum overhead is to be reached, too.
The further goals of the INTERBAS system design can be summarized as follows: maintaining physical integrity, increasing the level of data integrity by steps, access protection and multi-language understanding. Efficiency and robustness are substantial goals of the INTERBAS implementation design.

3. Principles of implementation

Taking into consideration above called aims, the following conclusions must be drawn:

Three-level architecture

The support of different user classes with a correspondending level of data independence causes consequent realization of a three-level architecture. That means, that all aspects of semantics of data base are isolated, on the one side, from aspects of physical implementation and, on the other side, from aspects of individual data applications. Therefore, the DBMS INTERBAS on principle distinguishs between

- external views,
- conceptual views,
- internal views,

directed to data or information.
According to different tasks and responsibilities of user classes an entry is possible on each architectonic level by means of user languages.

Coexistence of data models

The profile of user classes being differentiated in a great way forces to integrate the coexistence of data models in the INTERBAS project. The following three aspects should be considered:

1. The user can apply several external models and languages using his INTERBAS installation.
2. Different data models are available on different levels of architecture.
3. For an INTERBAS installation an alternative selection of a data model must be made on the conceptual and internal level, which is suitable for the correspondending level of requirements.

The decisions about the application of data models on each level on principle should be independent from the selected model of the other level.
All schemata required for the data base descriptions and their applications will be managed by an unified data dictionary system.
The data independence will be guaranteed by flexible mapping facilities for the schemata. The mapping schemata are to be located in the data dictionary.

Different control facilities

A great amount of different requirements determines the control of DBMS activities. Different control facilities such as traditional control modes, control of private data bases, concurrent requests to one data base, control of "long-term" transactions are necessary to build in according with the goals of DBMS installation, used computer, the place of DBMS installation in an hierarchical or distributed system.
The extent of above mentioned facilities is to be divided into subsystems in such a way, that they are

- either alternatively usable,
- or being able optionally to extend the facilities because of a higher level of requirements.

The user can figure his own concrete INTERBAS DBMS from these subsystems representing marketable products. Stable interfaces between these products form a substantial basis for a long-term open ended system.
All products will be developed in such a way, that they can be adapted to minimum requirements by means of generation and / or dynamic adaptation. The below presented architecture of INTERBAS products is concluded from the system goals and implementation design.

4. INTERBAS architecture

The INTERBAS architecture is illustrated in Fig. 1.
The following aspects determine the INTERBAS architecture:
- separating data management facilities from task control facilities
- structuring data management facilities on the basis of data base architecture and data models
- relative autonomy of managing meta-data.

The following products of the INTERBAS DBMS are planned in the first stage:

Products for data management

NWP - network processor
The network processor manages data and access paths. The underlaying data model is a modification of the CODASYL DBTG proposals. The most important data structure types are RECORD, SET and INDEX. The network processor also includes the internal data management facilities of the DBMS DBS/R.

FFP - flat file processor
The flat file processor manages table-like data. An index can be assigned to columns and groups of columns. The capability of the flat file processor is equivalent to one of the one-tuple-interfaces of existing relational DBMS. The flat file processor represents a subset of network processor.
Network processor or flat file processor provides a minimum of data management facilities within an INTERBAS installation.

BRP - basis relation processor
The basis relation procesor is the most simple facility for representing a conceptual schema.The SQL data base sublanguage is used for description and manipulation of relations.

EERP - extended entity-relationship processor
The extended entity-relationship model can be used as representation of conceptual schema in a higher level. This model is able to capture more of the meaning of applications.

VRP - view relation processor
The view relation processor allows describing and processing derived relations, which are more suitable for application programs or end users. The SQL language can be used, too.

QBE - Query-by-Example
The Query-by-Example facility allows screen-oriented queries for users. It realizes table operations, which can be used easily. Queries must be set by filling examples into patterns.

CAD/CAM interface
The CAD/CAM interface supports the development of CAD/CAM software by specifying specific data types and creating private, temporary data bases.

Each of the three external systems implicates the application of one of the conceptual systems.

Control systems (monitors)

MUMS - multi-user monitor system
The multi-user monitor system manages concurrent tasks, providing them data management facilities in a co-ordinate manner. It assures physical integrity and controls communication with users or virtual operating system.

SUMS - single-user monitor system
The single-user monitor system is a subset of the multi-user monitor system, realizing minimum requirements. It will also be designed for the application of private data bases in CAD/CAM work stations.

Data dictionary systems

DDM - minimum data dictionary
The minimum data dictionary provides a minimum of management facilities for description data (meta-data) about application data. The description data will be stored as relations, that mainly will be accessed by DBMS. It enables users to produce reports by writing SQL statements.

DDA - advanced data dictionary
The advanced data dictionary will be implemented as a comprehensive meta-data base, exceeding the capability to manage the description data.
Although above mentioned facilities of INTERBAS of the first stage are mainly compounded of internationally known facilities - here in a new system structure - the following outlooks should be stressed:

- Without restoring data base between several DBMS, INTERBAS processes the same data base using either relational operations or network ones.
- The efficiency of relational operations can be improved significantly, if the join operations are implemented by internal network structures.
- The flexible mapping facilities of INTERBAS nearly permit all efficiency-oriented changes of internal structure of data base without effects on conceptual objects. Therefore, the data independence will reach a higher level.

5. Conclusion

In regarding of the trends in the field of DBMS, the DBMS INTERBAS of the first stage meet only a defined amount of future requirements.
The presented data base and DBMS architecture are necessary prerequisite to pursuit of further goals. That is valid for both developers and users of perspective DBMS or Knowledge Base Systems.

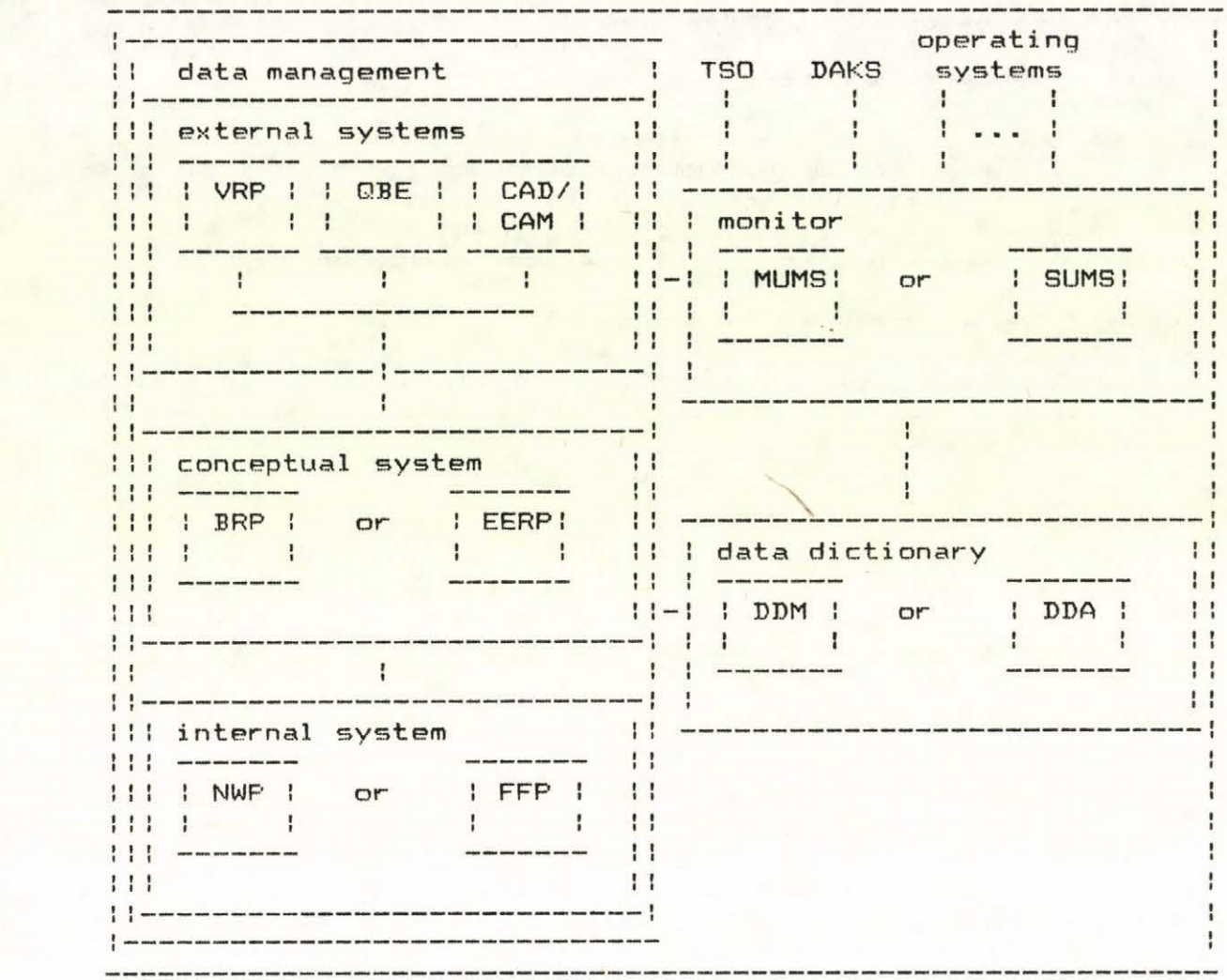


Fig. 1 INTERBAS architecture

VRP	view relation processor
QBE	Query-by-Example
CAD/CAM	CAD/CAM interface
BRP	basis relation processor
EERP	extended entity-relationship processor
NWP	network processor
FFP	flat file processor
TSO	time sharing option
DAKS	data communication system
MUMS	multi-user monitor system
SUMS	single-user monitor system
DDM	minimum data dictionary
DDA	advanced data dictionary

Literature:

Bittner,J.: Zur Aufgabenstellung des neuen Datenbankbetriebs-systems INTERBAS. Neue Tech. Buero (Deutsche Ausgabe) Berlin 32(1987)3, S. 78-79

MTA SZTAKI Tanulmányok 203/1988 p.121-134

LATOR - a Database Management System For Local Networks

László HANNÁK, Tibor REMZSŐ, Ferenc URBÁNSZKI

Computer and Automation Institute
Hungarian Academy of Sciencies

Basic Concepts

LATOR is a general purpose database management system for establishing and maintaining a database environment for a set of computers connected in a local area network. The data structures defined for the database are available to programs written in a host-language or, alternatively, they can be accessed by instructions given from key-boards directly. Consistency of the data structures is preserved automatically. Also it is made possible to the programmer to establish a deeper, 'semantic' consistency among the data, i.e. some kind of logical consistency not implied by the data structure itself.
The philosophy of LATOR is somewhat different from that of the commercial database management systems like dBASE or DATAFLEX etc., since its kernel is resident in a dedicated machine called SERVER: the requests and instructions coming from the terminal machines are executed on it by a program also called SERVER. The users communicate with this intelligent system whenever they deal with data structures known and supported by LATOR. This conceptual difference implies quite a lot of others, too, which facilitate the developing various types of end-user program packages. At the same time, LATOR does not support inexperienced people in developing their own systems but rather the professional staff in a software house. So the main application area of LATOR is thought to be a software house workshop where packages are developed by professional programmers.

Research partially supported by Hungarian National Foundation for Scientific Research, Grant No 1066

1. Data structures supported by LATOR

1.1.Data records. Acceptable to the LATOR system are data record-types identified by names. The number of occurrences of a record-type is practically unlimited, it is the storage capacity of the actual configuration only, which sets limits. The set of all occurrences of a record-type is called a (logical) file. The name of this file is identical with that of the record-type.An occurrence of a record-type, shortly: a record is identified by a serial number (starting from zero). Serial numbers are unique as long as the record is not deleted. The numbers of deleted records, however, can be given to newly inserted records.

Access to the data records on the file level would mean that a record in a file can be accessed, modified or deleted by giving its serial number.

A record-type is a sequence of fields. A name should be given to each field. The length of a record is essentially the sum of the lengths of its fields. The length of a record-type is not limited but all occurrences must have the same length.

Field-types are the following:
STRING - a sequence of arbitrary characters. (480 bytes at most, a character is one byte long. If, however, LATOR is used from PASCAL as host-language the length is limited to 255 bytes.)
INTEGER - signed integer number representable by 16 bits at most, in binary form.
LONG - signed integer number, 32 bit long at most.
DATE - a pseudo-string (e.g. 19861231) able to represent a date of type year-month-day from the 1st of January 1900 to the 30th December 2076.

Ordering is the only reason why types of fields should be distinguished at all. Otherwise the programs may interpret the strings in the fields according to their actual semantic role.

1.2.Sorted files, access by keys.
It is possible to access a data-record in a (previously defined) file by various keys defined in time of the database set-up. From the logical point of view the records will then be sorted in ascending order according to the defined keys. Each key should have a name given in time of

the database scheme description, and these names must be different from each other, as well as from the file names. A key is consisting of key-fields (or, with other words, of elementary keys) also having names, and each key-field should be a field of the respective record-type or a continuous part of such a field, but in the latter case the field must be of type STRING. Sorting is understood as follows:

-For elementary keys derived from fields of type STRING: the ASCII order. This ordering can, however, be overruled in time of the database definition. The redefined order will then be valid for all strings in the database.

-For elementary keys of type INTEGER or LONG: the usual ordering with due regard of the sign; negative numbers are always smaller than positives.

-For keys consisting of more than one elementary key-field, the ordering of the elementary key-fields among themselves can be fixed in time of the key definition.

Maintenance of the sorting is automatic: insertions do not destroy it. The usual operations for sorted files (see 1.4.2.) are implemented.

1.3. Relations.

The LATOR system is able to establish binary relations between data records. On the conceptual level a relation-type is a set of pairs of data records. Each relation-type must have a name. If a pair of records is an occurrence of a relation then we say that they are connected by that relation. The two records in a relation are called coordinates of the relation-occurrence.

When a relation is defined the record-types of the coordinates can be restricted but the restriction is not obligatory. If a restriction is declared then automatic checking takes place whenever a new occurrence of that relation is going to be inserted. It is also, however, possible that the first (or second) coordinate of a relation comes from different data files, as mentioned above.

Relations can be declared symmetric or asymmetric. If symmetric then the order of the coordinates is irrelevant, otherwise, however, because the inverse relation is also established obligatorily, a name should be given to that inverse,too. E.g. if an asymmetric relation-type is called CHILDREN then name PARENTS can, maybe, given to the inverse relation, which must be brought to existence .

Operations available on binary relations are the following:

-Connecting two records according to a given relation-type.

-Disconnecting former connected records.

-Checking whether two given records are or are not connected by a given relation-type.

-Given a record as first coordinate of a relation: search for the connected second coordinates (see 1.4.3.).

As regards relations there are operations automatically executed.

-If a data-record is deleted then all of its connexions are deleted, as well.

-If a connexion is inserted in an asymmetric relation then its inverse is established and inserted.

-If a connexion is deleted then its inverse is also deleted.

1.4. Navigation

Navigation is possible by currencies. On each terminal a program can be active and each of them has its own currency table maintained by the SERVER. The content of such a table is

-the serial number of the current record from each data file (record-type), or the value -1 if no record is yet current from that file,

-content of the current row of every relation-type: the serial numbers of the coordinate-records, as well their type.

1.4.1. Navigation by serial numbers.

Navigation within occurrences of a certain record-type (within a data-file) is possible by the following instructions:

-Let the first record of a file be current.

-Let a record of a given number be current.

-Let the next record be current.

'First' and 'next' refer here to ordering by serial numbers only. Insertion of a new record makes it current implicitly and automatically. Currency is not lost and does not change if the current record is being deleted for certain commands.

1.4.2. Navigation by keys

The subsequent operations always apply to a fixed file and a key defined for that file. 'First', 'last', 'next' are related to the sorting (in ascending order) by the key.

-Let the first (last) record be current.

-Let the record of prescribed key value be current.

Without going into details here it is mentioned only that it suffices to specify a certain part of the key value in an instruction. If no record exists with the specified key value then the next existing record becomes the current one.

-It is possible to get access to the key of the current record. (It is sometimes needed because a record could have been made current in another way, too.)

-Let the next (prior) record be current.

1.4.3. Currency in relations.

There is a kind of sorting automatically established in the relations, as well, in order to keep occurrences with identical first coordinates close together. So we can speak about the 'next row' in a relation. Currency is maintained in relations, too, and it is possible

-to make the first row of a relation be current,

-to make the next row be current,

-given the value of the first coordinate, to make the first row with that first coordinate be current.

It is also possible to navigate into a data file through a relation, because we can

-make current a data record, the first (or second) coordinate of which is current in a relation,

-have access to the content of the current relation.

1.5 Data access.

The set of operations available for user programs makes possible the reading of a record, inserting a new one or update it by listing the names of the fields of the relevant record-type. It is, however, not obligatory to read or write the complete record, the operations refer only to the fields actually involved. This approach has the advantage that, as far as the names of the file, those of the records and fields remain unchanged, the programs need not to be recompiled but can be used without modification if the database had to be restructured, e.g. extended by new fields.

It is possible to find a key value or elementary components of it. Also the content of the current relation can be obtained together with the record-types of its coordinates (whether or not the record-type of the coordinates is restricted).

2. The main parts of LATOR

The main parts are the following:
- Database definition
- Network server
- Utility programs
- Query system

2.1 Database definition

The database must first be described. The number of the data bases simultaneously present in a system is logically not limited. The definition phase is not interactive, there are no supporting instructions for that.

To be described are:
- -the name of the database
- -the logical data structures
- -the devices for storing the system
- -the logical order of the characters (see 1.2.)

As for definition of a database a 'logical scheme', a physical scheme' and an 'order of characters' should be given. There is a special language in form of a text-file for the definition. Its processing (translation) is the task of the program SCHEME. The utility program CHARS is provided for processing the requests for re-ordering the characters. The original text-file is kept for documentation. The result of the above mentioned processing is a database scheme in a form the SERVER program can make use of. During the processing of the descriptor text-file a checking is taking place, too, in order to avoid contradictions as much as possible.

The database scheme has the following parts:

-A table for the logical scheme: it contains the invariant elements of the data structures.

-A table of the physical scheme: names of the physical files the database itself are kept in it.

-A table with the time-dependent information about the database.

Because checking is taking place while the logical and physical description of the database is processed, the construction of the physical scheme is optional in order to make possible a first run for the syntactic checking only.

After having prepared the database scheme the environment the data handling algorithms suppose to exist, is ready. Now the database can be filled up with data coming from keyboards or, alternatively, from other sources, via batch programs, e.g.

In addition, beside of the definition of the database, certain other informations, called 'operation parameters' should also be given. Since they are scheme-independent they can be fixed or modified later.

2.2.The network data server.

In a network a dedicated computer should exist, as well as a program running on it, which is able to execute data manipulation procedures. This computer and the mentioned program are called, in what follows, SERVER. User programs on the terminals communicate with the SERVER by a network interface. The SERVER is constructed so that it does never put questions and wait for answers, it answers only to requests coming from the terminals. These must be correct,i.e. they must correspond to the logical scheme description. After the necessary transformation the instructions are executed and queries answered by the SERVER according to the point of view of the respective user-terminal, i.e. each program has its own currency table for navigation (see 1.4.).

The SERVER deals with the requests one by one, therefore they are queued. Too long queues are not likely to come into being since the simple LATOR instructions are executed rapidly enough. If it yet happens the request is rejected for a while and the 'guard' at the terminal repeats it later again.

There are two software layers ensuring the correct run of the users' programs. One is essentially a network interface which transforms the LATOR instructions for the actual network communication system. So it is possible to connect LATOR to any network communication software by adapting this layer. The other layer is, in fact, a language interface: it translates LATOR instructions and makes possible to write transactions in a host-language with requests in it for the SERVER. The interface translates the requests formulated on the logical level to those correctly interpretable by the

SERVER program. For the time being this interface has been written for TURBO PASCAL and MICROSOFT-C as host-languages.

2.2.1. Concurrency problems.

In a database environment it is highly important to avoid break-down situations where some parts of an operation have done already but not the rest of them. It is as well important that the necessary steps for avoiding such situations do not burden the users' programs but the database management system itself be equipped with preventive measures. Therefore all the elementary LATOR instructions (both for the data files and the other database structures) are made atomic, i.e. either all sub-operations are executed or nothing done.

There are two mechanisms provided for that. First, the server does not start executing a new instruction before having finished the previous one with or without success. Failure may be caused by an error of the user (reference to a non-existing record-type, e.g.), in which case an error message is sent to the terminal the respective instruction came from. If, however, the normal functioning of the SERVER is hindered by a hardware break-down, lack of space or other cause then execution is suspended, the entire LATOR system has to be re-started from a journal-file and a previous, still consistent status of the database created. The recovery procedure is described later.

Moreover the programmer may request that certain compound operations (e.g. transactions or parts of them, deletion of a record from a file and insertion to an other e.g.) should be regarded atomic. Such parts should start with a LOGON instruction and closed with a LOGOFF. The sequence between LOGON and LOGOFF will then be regarded atomic by the SERVER. The restoring mechanism of the journal-file is built up so that the result of a sequence of instructions between LOGON and LOGOFF is either kept completely or rejected altogether. (Note, however, that the structuring with LOGON and LOGOFF instructions can be made quite transparent in a program package made for an end-user.)

Algorithms of the recovery process.

The journal-file itself can and should be created by a utility program JRNCREAT in time of installation of the database. It will then exist for as long as the database itself.

When the SERVER is started running it is checked, first of all, whether or not the previous run had normal end. If not then the content of the database pages found in the journal

are re-written into the database (in the reverse order) so its end-state will correspond to the last instant it was still certainly consistent. If something goes wrong just in this phase the SERVER should simply started again. The user is meanwhile informed what are the programs (transactions) to be repeated again because their results have been lost by the recovery process.

Logging can be suspended temporarily supposed the former run of the SERVER was correct. Switching off may speed up long batch programs. But before doing this a complete copy of the database should be made and restarting, if needed, initiated from that copy, of course. An attempt to restart from the (suspended) journal may cause total confusion.

A long journal-file is, in fact, a nuissance because too many actions have to be repeated in case of break-down. But deleting the journal is possible only if there are no open transactions in progress and all database files are closed such that a complete consistent version of the database exists on the disk. On the other hand, file-closing operations are rather time consuming, even superfluous, because during normal run they must be re-opened again, anyway. Thus there is an adjustable parameter for setting limits to the allowable number of open transactions. If that number is reached then no new transaction is accepted for a while, the SERVER gets through all open transactions - if possible - then all files are closed and the existing journal deleted. The above described procedure is called check-point forming. The optimal value of the mentioned parameter, i.e. how often should check-points be formed may depend on the hardware configuration, as well as the habitual use of the database so it can, supposedly, be found by trials.

The command-file.

Logging and check-points are provided for avoiding the destructive effect of a power-supply failure or a break-down of the central processor. But if the disk goes wrong then recovery can only be started from a previously saved version of the database kept on a storage device other than that used normally by the SERVER. For making the restoring process more comfortable, it is possible to request from the SERVER the preparation of a so-called command-file. It is a sort of journal, for containing all update commands the SERVER has obtained since the last check-point. The command-file should, of course, be kept stored on another, independent storage device, too.

Concurrent access.
It may happen that two transactions want to get access to the same data record. For avoiding confusion it is possible to lock the current record by a LOCK instruction. Until UNLOCK other programs cannot lock it themselves, and they get access to it for reading only. A message is sent to the respective terminals by the SERVER that the record is locked and they must wait.

Deadlock situation may arise if there are programs running and more than one record is wanted to be locked by them. The SERVER is not prepared to resolve deadlocks automatically, therefore solution of the deadlock problem remains a task of the programmers who develop the end-user packages. For avoiding deadlocks the convention may be of help that the record-types should be fully ordered in an arbitrary but fixed way and the order of the locking instructions in the programs are agreed to correspond to the ascending order of the record-types.

Moreover, there can semaphores be defined, which can be set and reset, and this latter system does not depend on the locking of the records.

Parameters for the network utilization.
The run of the SERVER can be conditioned by a set of parameters, which should, by answering questions, be given in time the SERVER is activated. (A convenient set of parameter values can be put on a parameter-file and if nothing has to be changed then it suffices to make reference to this file at the start.)

The parameters are as follows:

-The number of terminals in the network to be served simultaneously.

-Size of central memory for file-management purposes (paging).

-The area reserved for users' programs. It can serve as a common area for transfer of parameters between programs so it remains untouched even if programs are exchanged.

-A switch whether logging is needed or not.

-Another switch for the command-file (whether it should or shouldn't be maintained).

-The size of the extension-unit in case the journal-file is full up.

-Number of transactions (those altering the content of the database) after which check-point forming is wanted (i.e. a full copy of the database stored on the disk and the journal restarted).

-A switch for tracing. If 'on' then the operations initiated from the terminals are traced. This option can be useful in time of program debugging.

-Test-regime switch. If it is 'on' then errors in a LATOR instruction (e.g. reference to a non-existing record-type) does not hinder the running of other programs, a message is sent only to the terminal the erroneous instruction came from. If the switch is 'off' then similar errors cause the stop of the SERVER, i.e. the entire system is suspended.

-The maximum number of simultaneously used locks and those of the semaphores.

-A switch whether the current record should be locked automatically.

-A further switch whether the restarted server should or shouldn't update the existing database by the instructions found in the command-file.

2.3. Utility programs.

2.3.1. Database dictionary generator (KONVERT and KONVERC)

This utility creates a dictionary of all names used in a given database. It is provided for making LATOR calls simpler. KONVERT serves for TURBO PASCAL as host-language, KONVERC for MICROSOFT-C.

2.3.2. JRNCREAT.

Creates the journal-file as well as the command-file.

2.3.3. UTESZT.

It makes possible for the database administrator the use of all LATOR instructions directly from a terminal.

2.3.4 Utilities for reorganization: SAVEREL, LOADREL,RELOAD, CHKONV.

The database administrator can, by these utility programs, reorganize the structure of the database, and fill up the new structure by the content of the old one. In particular, CHKONV creates a file, by which it is possible to convert character codes if a database has to be transferred to an other device where the character codes are different.

2.4 Intelligent query environment.

It supports the construction of procedures, which use the database in batch-like manner, e.g. for listing.

Features:

-Records satisfying certain conditions can, from a data file, be collected (the records selected in this way remain sorted).

-The collection of records chosen as above, can be submitted to a selection by giving further conditions.

The conditions can be constructed, by logical operations, from terms which mark out intervals from a key or other record fields. If certain records are given then those in relation with them can be found and listed. (The 'given' records might have come from a previous selection, e.g.)

-Listing and vizualizing the selected records.
-Resorting of collected (selected) records.

Queries should be formulated in the LQUERY language. Programs in LQUERY can be written, edited, translated and run in an environment similar to that of TURBO PASCAL.

LATOR - a Database Management System for Local Networks

Hardware and Software Requirements

Both the hardware and software requirements depend on the question whether LATOR is intended to be used on a single computer or in a local network of microcomputers.

1. Hardware

Hardware items needed for LATOR are the following:

- IBM PC/XT or AT or a microcomputer compatible with them.
- A floppy drive, at least one.
- Background store device for the programs and the database with capacity of 1 Mbyte about.

Further items for the local network:

- IBM PC or compatible microcomputers with 256 Kbyte memory each.
- LAN interface cards.

2. Software

General software:

- DOS operating system version 3.1 or higher.

- Network interface. Some kind of network software is certainly needed, moreover programs which connect LATOR with the network, e.g. in case of 10NET the TENNET.EXE program on the server machine, on the others a USERNET.EXE. Moreover, independently of the network, the MEGS.EXE program must be present on all machines that use LATOR.

- MISS.EXE. There are lots of working places where the database management system must be switched on all the day but, in fact, it is not needed often. In order to make use of the server machine for purposes other than executing requests to LATOR, the database monitor is functioning as a background program, while some other task can be executed in the foreground. To make possible this style of work a multitask environment is needed and this is realized by an operating system like layer, called MISS.

- LOCAL.EXE or LOCALS.EXE. These programs are needed on the server machine only. LOCAL.EXE is the suitable one, in general. The server machine itself can be used as a LATOR user, and with LOCAL.EXE it is able to represent a single LATOR user. If, however, LOCALS.EXE is in then the MISS partitons can be organized so as to represent different LATOR users. So this option (LOCALS.EXE) is there for software developing purposes because it creates an environment where concurrency problems of multiuser program packages can be tested on a single machine.

- LATOR.EXE or LHATOR.EXE. The database is managed by the programm called in the documentation SERVER, and this can be either LATOR.EXE or LHATOR.EXE. The former is somewhat smaller and more rapid but is able to manage databases only, the logical scheme of which is not too large. LATOR or LHATOR should be present on the server machine only.

- DBEXIT.EXE. This program is there for terminating LATOR and close the database files correctly. Its presence is needed on the server machine only.

- There are, of course, some programs needed for communication with LATOR. Direct commands are transmitted by the UTESZT utility. The LATROQ query system or some end-user package written in a host-language (MICROSOFT C or TURBO PASCAL) can play this role.

LATOR is delivered on a floppy disk with the files:

MISS.EXE
LOCAL.EXE
LOCALS.EXE
LATOR.EXE
LHATOR.EXE
DBEXIT.EXE
SCHEME.EXE
MEGS.EXE
JRNCREAT.EXE
CHARS.COM
PROC.PAS
KONVERT.COM
PROC.LIB
KONVERC.EXE
UTESZT.EXE
SAVEREL.COM
LOADREL.COM
RELOAD.EXE
CHKONV.COM

From this collection a suitable combination sholuld be chosen.

MTA SZTAKI Tanulmányok 203/1988 p.135-148

DBASE PRODUCTS POSSIBILITIES COMPARISON FOR INFORMATION SYSTEMS ENGENEERING

Rumjana Kirkova, Juliana Peneva

Institut of Mathematics with Computer Centre
Bulgarian Academy of Sciences

1090 Scfia, P.O.Box 373
Bulgaria

I. INTRODUCTION

Recently fast development of microcomputers and their application in various fields is noticed.Corresponding software tools are being created with respect to microcomputers specific restrictions such as: secondary storage type,memory capacity,CPU - possibilities.There are many products today which are database management systems(DBMS) based on the relational model.They take into account the possibilities of microcomputers offering at the same time powerful data manupulation operations.In addition these products are systems that place an experienced user in full control with enough programming power and flexibility to design its own business applicatoins.

Among all existing programmable relational databases the following can be mentioned because of their wide-range possibilities and effectiveness(1):

- R:base Series 5000 (2)

The system includes own procedural language, a compiler, an easy-to-use application generator and a sophisticated import module.

- DataEase

It is one of the few systems which offers power and versatility combined with ease of use for less experienced user.

- KnowledgeMan (3)

This is an integrated system that uses its query language to establish relations relations between data files.It has a full-featured word processor, a business graphics package, a report painter and a communication package.KnowledgeMan is appropriate for professionals.

- Oracle (4)

This is a first-class system for professional database systems which implement a SQL query language.The user's interface is very good and a C-precompiler gives C-programmers a direct access to databases.Oracle is completely portable accross all type of machines: mainframes,minis and personal computers and is compatible with IBM's mainframe SQL/DS and DB2 systems.

- Paradox

This is a highly flexible database manager that gives users the possibility of choice an interactive mode or a procedural language called PAL.

- dBASE series

These products (dBASE II,dBASE III,dBASE III Plus) have become standard by which all other similar products compare themselves(5). Nowadays they are widely distributed because of their good possibilities and easy manipulating.They possess a simple command set appropriate for an end-user to process immediately its necessary information including at the same time procedural operators and structuring programming methods.Thus, the professionals can develop rather complex routines with considerable productivity.The success of dBASE series is due to the large variety of possibilities which are offered to the user depending on its training and knowledge.

Here we analyze the three versions of dBASE comparing some basic parameters.The possibilities and effectiveness of each product are examined.Some suggestions for their applicability in information systems engeneering are given.

II. DBASE II - A STANDARD FOR MICROCOMPUTER DATABASE MANAGEMENT

DBASE II is still used for the design of different systems independently of the existing new versions with larger possibilities.DBASE II is capable to handle tasks which up to now have been performed exclusively with minicomputers.Most important dBAE II features can be considered in the following directions(6):

1. Data Entry

DBASE II has no high-level language.Users screens can be created through the data manipulation language.There are good possibilities for including the necessary information and validity checking.Data entry fields appear in reverse video.However a significant disadvantage is that the result of the screen design can be obtained only after executing the program.

2. Data Manipulation Operations

DBASE II possesses the basic file operations but some of them are with considerable restrictions(6).For example, to create a new file is possible only if data is entered in a dialog fashion and a modify operation has as a result all data destroyed.All record operations are also on hand.Good possibilities for specifying different searching criteria exist.However they are limited by the weak sorting and indexing capabilities of dBASE II.The procedures are extremely slow and only a single key can be used.Multiple indices on single file are allowed but only one index at a time can be used.The relational operators:join, select,project are available.

3. Allowable Data Types

DBASE II has only three data types:character,numeric, logical.No use of arrays,date fields and user defined or virtual fields is possible.

4. Security

There aren't tools for file or different field security. Records cannot be locked, no password are supported.

5. Report Generation(7):

DBASE II offers the following good capabilities for report generation:

- breaking up character strings at a space if they are wider than the column
- automatic numbering and dating
- specification of column headings

However the change of a report description is rather difficult.There are tools for specifying subtotal fields and conditional reports.Statistical quantities are not provided.

6. Ability to Accept Data from Other Programs

MULTIPLAN and SUPERCALC can use dBASE II data files(8). There are ways to accept output data from other relational DBMS.Properly presented ASCII files can also be processed.

As a conclusion we can list the following inconveniences of dBASE II:

- the limit of 32 fields per record which lead to problems with larger applications
- the maximum of 1000 character record lenght
- the limit of 64 memory variables
- only two databases opened at a time
- a new record can be edited only while it is on the screen; after entering the last field there is no backing up to make corrections.
- difficult setting up and mantaining relationships among open databases; a common file is usually required to pass the related information from one database to another.
- difficult data managing
- poor sorting and indexing capabilities
- no value checking

However the following basic advantages of dBASE II are to be kept in mind:

- modest hardware requirements
- easy training

A more versatile adaptation of dBASE II is PractiBase(1),from Probase Group Inc.

III. DBASE III - AN IMPROVED VERSION OF DBASE II

DBASE III is the successor to dBASE II which realizes the full potential of the personal computer(IBM PC or compatible). Most of the dBASE II inconveniences are removed by developing new tools which facilitate its usage.These tools concern:

1. Data Entry(9)

A full screen text editor is included in dBASE III.So, various custom-designed screens are possible depending on the particular application.The system also offers default screens which format consists of fields aligning vertically down the screen and the amount of spaces indicated in reverse video. Thus, data entry errors are strongly reduced. The design of customized screens requires written procedures.It is not difficult because of the various editor possibilities.

2. Data Manipulation Operations (10)

A rich set of commands,both for interactive use and inclusion in data manipulation procedures, exists.The basic file operations are renovated removing the tipical restrictions for dBASE II.To define different searching criteria is possible using the command SET FILTER TO. The only limitation is the command line lenght(up to 254 characters). DBASE III has a powerful command REPLACE which allows data replacing in several files simultaneously.The multilevel Sort operation is particularly efficient allowing all exept free text key fields. There is no restrictions on the number of indices for a given file.The basic relational operators are improved.Unlike dBASE II the number of opened at a time files could be up to 10. Each of the opened files is connected with its own area.However the JOIN operation can work only with two open files simultaneously.The relationship between databases is set up and mantained without additional programming using the command SET RELATION TO. The concept of Aliases is introduced. So, multiple data files can be easily handled.

3. Data Manipulation Language

The language is powerful,versatile and has structured programming features such as:procedures,DO WHILE - constructions,etc. Additional functions (exponentiation,square root,logarithms) are supplied. They contribute to the wider language usage for many business applications

4. Allowable Data Types

All data types exept user defined fields and arrays are supported.Two new data types are included:

- Memo data type

It allows the inclusion of long text strings(up to 4000 characters) in a database.The text is actually stored in a separate file while the memo field hold the information of its location.This type of fields can not be indexed or searched.

- Date data type

This new type automatically performs a validation check on the month and day.Mathematical operations and specialized functions on it are also possible. The field size is set at eight characters and the format is DD/MM/YY.

5. Report Generation

The dBASE III report definition program is much easier for use than that of dBASE II. Report forms can be modified. A new non-columnar report format called "mailing labels" is created. It can also be used for other short forms such as checks.

6. Access to DOS Commands and Other Software

Unlike dBASE II, dBASE III can be connected with almost all software packages (11).:

- word processing packages - WordStar,Microsoft Word
- spreadsheets - Lotus 1-2-3, Multiplan
- DBMS - Symphony, Framework
- dBASE II - by means of the utility dCONVERT

The operation system can be accessed within dBASE III if at least 272K RAM are available and the DOS COMMAND.COM file is on the same disk.

7. Programming Techniques (12) .

New programming possibilities are added to dBASE III. They allow efficiency in command files and increase the speed particularly for developing large custom software systems. The most important of them are:

- procedure files and parameter passing

All repetetive functions and routine tasks in a given system can be combined in one procedure file as different command files (procedures). When a procedure file is opened all procedures (up to 32) within are stored in RAM and thus accessed much faster. Additional flexibility is obtained by the possibility of parameter passing. Each

procedure may contain any number (or none) of parameters. This permits the information passing without the use of specific memory-variable names.

- public and private variables

The dBASE II variables are global to all programs i.e. once created they can be only explicitly changed or eliminated with a RELEASE command. On the contrary in dBASE III the memory variables are automatically released after the command file execution. However if they are declared PUBLIC they can be available to all programs in a system and can not be erased after the end of a command file.

- date programming

Unlike dBASE II, dBASE III can sort records using a given date field, taking the year in consideration. It is possible to give a range of dates to search for by setting up a filter command.

8. Mathematical Capabilities

DBASE III offers many new commands, functions and operators for handling numbers such as: AVERAGE, ROUND, SQRT, etc. Some basic arithmetic on Date fields and Date type memory-variables is also available.

9. System Customization and Conversion

The full advantages of dBASE III's capabilities can be obtained by setting up special files to configure the system and by presetting some specific dBASE parameters.

- configuring DOS for dBASE III

The dBASE III software has the potencial to handle up to 15 open files simultaneously. To escape the DOS defaults to a setting that allows only eight open files at a time, a special file named CONFIG.SYS has to be created. It should contain the DOS commands:

files = 20

buffers = 15

which permit the usage of 20 open files simultaneously with 15 buffers. The CONFIG.SYS file is executed from DOS during boot-up. So,the full capability of the product can be used.

- configuring dBASE III for the user's needs

Each user can preset some dBASE III parameters according to its own needs and specific applications. These parameters are: screen color, function key commands, the default disk drive for databases, the prompt, the word processor, and so forth. To do this a specific file named CONFIG.DB is created. It contains any parameter which the user would like to set.

In addition to the improvements in the directions listed above, dBASE III offers several new commands and general-purpose functions.

Let us mention some dBASE III inconveniences:

- there is no possibility to correct the command line if once it is entered (its lenght can be up to 254 characters).

- there is no interactive debugging tool i.e. while a procedure is running no command lines can be entered.

- some hardware and software requirements are considerable increased (see Table 1).

Version Parameter	dBASE II	dBASE III
RAM	128 K	256 K
Number of floppies	1	2
DOS	PC DOS 1 or 2 CP/M - 86	PC DOS 2.0 or higher

Table 1

IV. DBASE III PLUS - POWER TO USER

This is the last product of the series which is known. It has many new features: interactive Query mode, expanded error trapping and debugging capabilities, improved report generator(13). Some of dBASE III Plus elements are:

1. Advanced Query System

It allows any user to build complex queries by selecting from pull-down menus. The results are obtained much faster than by the classical dBASE III method for defining queries through a corresponding language.

2. Applications Generator

This is a complex tool oriented toward non-professionals by which they can solve themselves their own problems.

3. Screen Painter

It represents a screen design feature for creating both a special screen file that can be directly used and standard dBASE code which

can be modified. Up to 16 linked screens can be defined thus eliminating the need to place many fields on one screen. So, the output data are more readable.

4. View Possibilities

They permit to set up relationship between different databases.

5. Local Area Network Facilities

This is the most important enhancement to the users interface. IBM/PC Networks and Novell Advanced Netware/86 are supported. New commands are included for locking files and individual records to prevent simultaneous access by more than one user. There are eight levels of security which are protected by passwords.

The possibilities mentioned above facilitate the users considerbly in their work with dBASE III Plus. At the same time they do not require specialized knowledge. That is why dBASE III Plus can be related to the Feurth Generation Languages.

In addition to these features dBASE III Plus has a two times faster sorting and ten times faster indexing. More than 50 new commands and functions are included which improve considerably the package possibilities. Simultaneously all dBASE III programs run under Plus with no changes.

V. COMPARISON OF THE DBASE PRODUCTS

The following criteria can be used for the dBASE products comparison:

1. Data limits

Feature	dBASE II	dBASE III(Plus)
Number of fields per record	32	128
Maximum characters per field	254	254
Maximum characters per record	1000	4000
Number of records/database	65 535	1 billion
Number of files open	2	10
Number of precision	10 digits	15 digits
Number of memory variables	64	256

2. Data Types

Feature	dBASE II	dBASE III Plus
Character	YES	YES
Numeric	YES	YES
Integer	NO	NO
Floating-point	YES	YES
Money	NO	NO
Logical	YES	YES
Date	NO	YES
Long text	NO	YES
Arrays	NO	NO
User defined	NO	NO

3. Data Manipulation

Feature	dBASE II	dBASE III Plus
Indexing	YES	YES
Number of index files	7	7
Compound indexes	YES	YES
Unique index values	NO	NO
Respecify index fields	YES	YES
Respecify file definition	YES	YES
Sorting	YES	YES
Number of Sort fields	1	7
Ascending order	YES	YES
Descending order	NO	YES
Multiple record deletions and updates	YES	YES

4. Processing Times (min:sec.hundredth-sec)

They are obtained by using a database with 1000 records on IBM XT with 512 K RAM (11).

Process	dBASE II	dBASE III
SORT ON one field	6:08.15	3:23.74
INDEX ON one field	2:59.39	2:04.01
INDEX ON three fields	5:27.73	3:08.72
REPLACE with active indexes	2:25.25	0:43.83
APPEND with active indexes	0:03.31	0:00.59
REINDEX (2 index files)	8:44.89	4:28.15
SUM FOR (10 records)	0:19.67	0:15.94
COUNT FOR (10 records)	0:19.63	0:15.19
PACK (no index file)	0:44.47	0:45.25
PACK (2 index files)	8:57.43	5:00.75
COPY (no index file)	0:59.39	1:47.82
COPY (1 index file)	2:14.11	2:16.57

5. Input Facilities

Feature	dBASE II	dBASE III(Plus)
Screen painting	NO	YES
Automatic screen definition	YES	YES
Programming screen definition	YES	YES
Number of files per screen	2	10
Number of screens per file	1	16

6. Output Facilities

Feature	dBASE II	dBASE III(Plus)
Arithmetic functions	YES	YES
Aggregate functions	YES	YES
Statistical functions	NO	NO
Date functions	NO	YES
Multiple file reports	NO	YES
Mailing labels	NO	YES
Screen output	YES	YES
Printer output	YES	YES
Disk output	YES	YES

7. Special Possibilities

Feature	dBASE II	dBASEIII(Plus)
DOS 2.0 directory support	NO	YES
Customize keyboard	YES	YES
Configuring dBASE	NO	YES
Macros	YES	YES
Assembler interface	YES	YES
C - interface	NO	NO
FORTRAN - interface	NO	NO
Proprietary high-level language	YES	YES

VI. APPLICATION POSSIBILITIES

The dBASE series products possess various possibilities and powerful tools which make them capable to be basis for the generation of different users applications. However some their peculiarities have to be considered when the corresponding business application is designed.

dBASE II

In spite of its restrictions dBASE II can handle a big quantity of various tasks. We know that recently the product is used by more than 1000 firms which produce business software. Two basic shortcomings are to be taken into account before developing a new application:

- the product is lacked of a good Sorting
- dBASE II is relatively slower than other relational DBMS because of the often accesses to the disk drives.

To overcome these incorveniences several support programs were developed. A new routine, called dSORT, is created to run within dBASE II or directly from DOS. It can sort up to 32 fields with any combination of ascending or descending order and it is ten times faster. A program generator dGEN is also available. Nevertheless dBASE II is suitable for design of systems for which the short responce time is not a decisive condition and a relatively small amount of data is processed.
Examples are:

- accounting packages
- inventory systems
- banking systems

dBASE III

The considerable improvement of dBASE III allows various business applications generation (15). Owing to the integrated full screen text editor and the inclusion of free text (memo) fields the development of the following applications is possible:

- text oriented data retrieval systems
- annotated bibliographies
- "facts management" systems
- office automation systems

As a conclusion, we can mention that the combination of the screen editor with the internal programming language leads to design of sophisticated aplications such as:

- contract generators
- technical manuals
- instruction curricula
- procedures for calculating financial data
- statistical packages
- debugging tools
- modifying existing software
- check writing procedures

REFERENCES

1. Dickinson J.(editor) "Programmable relational databases",PC Magazine, June 24,1986
2. Simpson A. "Understanding R:base 5000",Sybex Inc.,1985
3. Hecht M. "File and data base management systems for the IBM PC", Willey & Sons Inc.,1984
4. Csaba J. "Microcomputer and relational database management systems: a new strategy for decentralizing databases",Data Base,16,1,Fall 1984
5. Neary M. "DBASE III:The right stuff?",Computer Decision,July 15,1985
6. Brarucha,Kerman D. "DBASE II - A comprehensive user's manual", Tab books Inc,1985
7. Cassel D. "DBASE II simplified for the IBM personal computer" Prentice-Hall Inc.,1985
8. Badal D.,Reive R. "Database management in the microworld - dBASE II and MDBS III ",ACM's Database Week; Special Session on Databases for Business and Office Applications, May 1983
9. Brown C. "Essential dBASE III", Wadsworth Publishing Co.,1985
10. Prague C. ,Hammit J. "Programming with dBASE III" Books Inc.,1985
11. Simpson A. "Advanced techniques in dBASE III", SYBEX Inc.,1985
12. Carrabis J. "DBASE III advanced programming", Que Corporation,1985
13. "Programming with dBASE III Plus",Ashton-Tate, 1985
14. "Networking with dBASE III Plus",Ashton-Tate, 1985
15. Baker R. "Advanced dBASE III applications", TAB Books, 1985
16. Townsend C."Mastering dBASE III: a structured approach",SYBEX Inc.,1985
17. Rob P. "Programming with dBASE II and dBASE III",Wadsworth Publishing Co, 1986
18. "Learning and using dBASE III Plus", Ashton-Tate, 1985

MTA SZTAKI Tanulmányok 203/1988 p.149-158

DATABASE SYSTEMS AND THE QUERY OPTIMIZATION PROBLEM - ANALYSIS, TECHNIQUES, POSSIBILITIES

Dr. Rumjana Kirkova, Juliana Peneva
Institute of mathematics with Computer Centre
Bulgarian Academy of Sciences
Bulgaria, 1113 Sofia, Acad.G.Bontchev, bl.8

I. INTRODUCTION

Recently the database management systems (DBMS) become a standard tool designed to improve the creation of various end-user applications, in order to facilitate their access to the stored data. They offer a set of rather powerful operations and good opportunities to describe the received information by query. The query can be expressed in a number of different equivalent representation forms. That is why a reasonable question arise: which of the diverse users requirement expressions achieve acceptable performance or which query is optimally determined. Our opinion is that the problem is important for the relational systems because relational languages are at sufficiently high semantic level and optimization is feasible.

II. THE QUERY OPTIMIZATION PROBLEM: objectives, approaches, general framework

The term "query optimization" has two principal points to be clarified:

- it refers to the expression to be optimized as a query in spite of its origin: direct request by the end-user who need information about the structure or content of the database or transaction that change the stored data based on their current values;

- initially there is no guarantee that the chosen technique for implementing the query is actually optimal but surely there is an amelioration of the original version.

For this reason all strategies intended to improve the efficiency of query evaluation procedures concern query optimization. Its main objectives are:

- minimization the response time for a given system environment
- reduction the cost of technical resourse usage.

According to [1] the total cost to minimize is obtained by summarising:

- the secondary storage access cost;
- the storage cost;
- the computation cost;
- the communication cost.

Let us consider a simple example to illustrate the need and also some of the possibilities of the query optimization: The relational schema of the department-and-employees database is (key attributes are underlined):

EMP (EMP#, ENAME, DEPT, SALARY)

DEPT(DEPT#, DNAME, DIVISION, BUDGET)

Let us assume that the query "Get names of the employees, department, name and division which salary is greater than $2000" has to be executed.

Let us suppose that there are 2000 employees working in 10 departments of 3 divisions but only 200 of them have salary greater than $2000.

Let us examine the following two strategies:

A1. Compute the Cartesian product of relations EMP and DEPT . This step involves reading 2010 tuples, constructing a relation with 2000 x 10 = 20 000 tuples and writing them back onto disk;

A2. Restrict the result of step A1 to those tuples for which SALARY > $2000 . This step involves reading of 20 000 tuples but produces a relation consisting of only 200 tuples which can be kept in the main memory;

A3. Project the result of step A2 over ENAME, DNAME, DIVISION to produce the final result.

B1. Restrict the result of step A1 to those tuples for which SALARY > $2000. This step involves reading of 2000 tuples

but produces relation with only 200 tuples which can be kept in the main memory;

B2. Join the result of step B1 to relation DEPT over DEPT . This step involves the retrieval of only 10 tuples. The result is kept in the main memory again;

B3. Project the result of step B2 over ENAME, DNAME, DIVISION to produce the final result.

It can be easily calculated that in the second procedure the number of secondary storage accesses is 20 times less that those of the first.

This simple example demonstrates the overall query optimization problem is quite complex. Two main approaches are usually used to solve it: BOTTOM-UP and TOP-DOWN .

The current trend seems to be the TOP-DOWN approach. Using it the following framework of a general query optimization procedure can be proposed [1,2] :

1. Query representation
2. Query transformation
3. Query component evaluation
4. Choice of the cheapest query plan

III. THE BASIC STAGES OF THE QUERY OPTIMIZATION PROCESS

1. Query representation: The first step in query processing is to convert the query in some internal form. This can be done in a number of ways but the chosen formalism should represent all possible queries for a given language and to be neutral refering to the optimization strategies. For our purposes we will use the relation algebra. Precisely the syntax can be defined by the following BNF grammar [2] :

```
rel_defn ::= DEFINE RELATION rel_name [attr_n_list]
selection ::= primitive WHERE selection_pred
primitive ::= rel_name(expr)
projection ::= primitive|primitive[attr_spec_list]
attr_spec ::= attr_name|rel_name.attr_name
```

infix-spec ::= projection infix_op projection
infix_op ::= UNION | INTERSECT | MINUS | TIMES | JOIN | DIVIDEBY

2. Query transformation: Obviously a query can be expressed in a number of different ways and in addition some semantically equivalent expressions may exist, even within a given language. That is the reason to convert its representation into more efficient form applying a set of well-defined logical rules. The query transformation process has three stages:

a) standartization - its goal is to construct a standartized starting point for query optimization throught a normalized version of the query internal representation form [4] This form can be achieved using the following rules:

COMMUTATIVE RULES	A AND B $\Longleftrightarrow$ B AND A
	A OR B $\Longleftrightarrow$ B OR A
DISTRIBUTIVE RULES	A OR (B AND C) $\Longleftrightarrow$ (A OR B) AND (A OR C)
	A AND (B OR C) $\Longleftrightarrow$ (A AND B) OR (A AND C)
ASSOCIATIVE RULES	(A OR B) OR C $\Longleftrightarrow$ A OR (B OR C)
	(A AND B) AND C $\Longleftrightarrow$ A AND (B AND C)
IDEMPOTENCY RULES	A OR A $\Longleftrightarrow$ A
	A AND A $\Longleftrightarrow$ A
	A OR FALSE $\Longleftrightarrow$ A
	A AND TRUE $\Longleftrightarrow$ A
	A OR TRUE $\Longleftrightarrow$ TRUE
	A OR NOT(A) $\Longleftrightarrow$ TRUE
	A AND NOT(A) $\Longleftrightarrow$ FALSE
	A AND (A OR B) $\Longleftrightarrow$ A
	A OR (A AND B) $\Longleftrightarrow$ A
	A AND FALSE $\Longleftrightarrow$ FALSE
DE MORGAN'S RULES	NOT (A AND B) $\Longleftrightarrow$ NOT(A) OR NOT(B)
	NOT (A OR B) $\Longleftrightarrow$ NOT(A) AND NOT(B)
DOUBLE NEGATION RULE	NOT (NOT (A)) $\Longleftrightarrow$ A

Using these rules the following transformation is possible:

WHERE a OR (b AND c) $\Longleftrightarrow$ WHERE (a OR b) AND (a OR c)

the predicate is in conjunctive normal form.

The relational algebra expressions are high level and symbolic. Therefore they can be also easily manipulated with other transformation rules:

(A JOIN B) WHERE restriction on B $\Longleftrightarrow$
(A JOIN (B WHERE restriction on B))

(A JOIN B) WHERE restriction on A AND restriction oh B $\Longleftrightarrow$
(A WHERE restriction on A) JOIN (B WHERE restriction on B)

b) simplification - the main objective of this stage is to eliminate some redundant expressions. They can be simplified using the Idempotency rules. For example

WHERE (DEPARTMENT = 'ADMINISTRATIVE') OR
(BUDGET > '2000' AND BUDGET > '2000') $\Longleftrightarrow$
WHERE DEPARTMENT = 'ADMINISTRATIVE'

If the semantics of the comparison operators are explicity taken into account some simplifications can also be obtained. Consider the expression

A.F1 $\geqslant$ B.F2 AND B.F2 > C.F3 AND C.F3 $\geqslant$ A.F1

which can be replaced by the Boolean value FALSE because of the contradiction A.F1 $\geqslant$ A.F1

Another application is the constant propagation based on the transivity lowes

A.F1 > B.F2 AND B.F2 = 20 $\Longleftrightarrow$ A.F1 > 20

c) amelioration - Usually query simplification does not result in a unique expression. Several semantically equivalent and nonredundant expressions for a given query may simultaneously exist but they strongly differ from performance parameters. The construction of improved with respect to evaluation performance expressions in the goal of amelioration. Usually it is based on query transformation heuristics. There are here some simple rules and techniques:

- a restriction of a projection is equivalent to a projection of a restriction i.e.

(A [attr_list_1]) WHERE restriction 1 $\Longleftrightarrow$
(A WHERE restriction 1) [attr_list_1]

- a sequence of restrictions can be combined into a single restriction i.e.
  (A WHERE restriction 1) restriction 2 ... $\Longleftrightarrow$
  A WHERE restriction 1 AND restriction 2 ...
- a sequence of projections is equivalent to the last projection only i.e.
  (A[attr_list_1])[attr_list_2] ... [attr_list_n] $\Longleftrightarrow$
  A [attr_list_n]
- the selective operations restriction and projection have to be moved over the constructive operators join and Cartesian product [5]
- query detachment

Obviously the ameliorating transformations use complex information about the query itself, the relational data structures and general heuristics rules. A consideration of some integrity constraints is also possible.

3. Query component evaluation: After having converted the internal query representation into the most appropriate form, the query expression has to be evaluated. The basic strategy is to consider it as a sequence of comparatively low-level operations such as: restriction, join, projection etc. This means that the query is mapped into alternative series of elementary operations for which a set of implementation procedures with their associated costs exists. The cost implementation of a single operation is heavily influenced by the following factors:

- the existence of indexed or other accessed paths
- physical clustering of records
- the support of sorted relations
- distribution of the stored data values
- the current state of the database.

As a final result a set of low-level procedures with an associated cost measure for each query expression element is obtained. The components of this set are used for creating different query plans.

There are many methods for implementing the major relational operators [6,7,8,9,10] . The following basic techniques can be mentioned:

- NESTED LOOPS : all possible combinations are inspected; for the join-operation it can be described in pseudocode as follows

```
                do  i=1,n1
                    read i-th element of rel-1
 outer          |   do j=1,n2
                |      read j-th element of rel-2
                |      join the scanned tuples if the joim-
 inner          |           condition is satisfied
  loop          |   end
 loop           |
                end
```

- SORT/MERGE METHOD : it is based on the order in which relation elements are accessed.
- HASHING : first the relations are hashed on the attribute used values and after that scaned using the created hash-tables.

4. Choice of the cheapest query plan: The final result of the query optimization process should be an efficient plan which is built by combining together one implementation procedure for each low-level operation inthe query expression. The following three basic stages can be mentioned:

a) generation of all reasonable logical query plans: A logical query plan describes a sequence of operations or intermediate results leading from the existing relation to the final query result. In fact there are many possible combinations and the generated set of query plants have to be restricted within certain bounds. Otherwise the optimization effort can't be kept acceptable. There are different approaches to the problem some of which are heuristics. An opportunity is to use a rigid set of query transformation rules [5] and to generate exactly one plan that need not be optimal. The opposite method [11] create all nondominated possible plants in a

given physical environment but this can result too expensive for very complex queries.

b) cost analysis of the generated query plan using some details of the physical data representation. The final cost measure is the number of secondary storage accesses but the size of the intermediate results has to be taken into account. It is extremely difficult to estimate the relationship between these two factors but in essence it depends on the physical storage structures and the number of accessed elements.

c) selection of the optimal query plan: In this stage the cheapest query plan has to be selected using the cost estimates. Two basic strategies to solve this problem can be considered:

- determine the costs of each alternative query plan [11,12]. This approach includes parallel processing of query components to avoid repeated access to the same data and the so called "feedback" method which uses partial results of a join-operation in order to restrict its output. A disadvantage of this technique is that the optimization effort is not able.
- compute the cost of the strategies incrementally in parallel to their generation [13]. This allows all strategies with common part to be evaluated together and so to reduce the optimization costs.

Accurate usage of cost estimates and therefore selection of realy good query plan is a difficult problem.

IV. OPTIMIZATION IN SOME DBMS

1. System R: This is a compilling system and thus the optimization problem is actual because each query expression may be optimized once and after that executed as time as necessary. A query is expressed in SEQUEL and consists of a set of SELECT-FROM-WHERE-blocks, some of which may be nested. The optimization strategy is the following:

- choice of the block order; if the blocks are nested they are performed according their nested order i.e. as specified by the user
- optimization of each block - its cheapest implementation is used

For a given query block there are two cases to consider [14]:

- the block involves just restriction and/or projection on a single relation. In this case to choose a strategy for costructing it, the following information is used:
  a) statistical data from the system catalog such as: number of tuples in each relation, number of pages occupied by each relation, number of distinct data values for each index;
  b) formulas for costs of low-level operations and for size estimates of intermediate results.
- the block involves two or more relations to be joined together with probably local restrictions and/or projections. In this case the following strategy is used:
  a) treat each relation as in first case;
  b) determine the sequence for performing the joins.

Any particular join is implemented by nested loop or sort/merge method depending on their costs 8 .

2. INGRESS : The basic idea in this system is to reduce a tuple query with multiple tuple variable into a series of smaller queries each of them with one such variable only. This general strategy is called query decomposition. The techniques to achieve it are two:

- query detachment ;
- tuple substitution - this is the process of substituting for one of the variables in the query a tuple at time.

First query detachment is applied until the query can not be decomposed via this technique any further. After that it is processed by tuple substitution.

There are different algorithms for reduction the irreducible components and for choosing the variable for tuple substitution [15] . The optimization strongly depends on the latter chäice. INGRESS usually tries to choose the relation with the smallest cardinality for the tuple substitution. As in system R statistical information kept in the system catalog and partucular access paths (e.g. a hash or an index) are also used for scanning relations.

R E F E R E N C E S

1. Mattias Jarke, J.Koch "Query optimization in database systems" ACM Computing survey, 16, 2, 1984
2. J.Date "An introduction do database systems", 1986
3. E.F.Codd "Relational completeness of database sublanguages", in Courant Computer Science Symposia N6:Database systems 1972
4. W. Kim "On optimization an SQL-line nested query", ACM, Trans. database systems, 7, 3, 1982
5. J.M.Smith, P.Chang "Optimization the performance of a relational algebra database interface", Communication ACM, 18, 10, 1975
6. R.M. Pecherer "Efficient evaluation of expression in a relational algebra", Proc.ACM Pacific Conference (April 1975)
7. L.R.Gotlieb "Computing joins of relations", Proc. 1975 ACM SIGMOD Intern.Conference on Management of Data (1975)
8. M.W.Blasgen, K.P.Eswaran "Storage and access in relational databases", IBM System Journal, 16, 4, 1977
9. T.H.Merett "Why sort/merge gives the best implementation of the natural join", ACM SIGMOD, Record 13, 2 (January 1983)
10. D.J.de Witt and al. "Implementation techniques for main memory database systems", Proc.1984 ACM SIGMOD Intern. Conference on Management of Data (June 1984)
11. S.B.Yao "Optimization of query evaluation algorithms", ACM Trans. database systems, 4, 2, 1979
12. M.W.Blasgen, K.P.Eswaran "On the evaluation of queries in a relational database system", IBM Research Report 1745, IBM Research Lab., San Jose, California, 1976
13. A.Rosenthal, D.Reiner "An architecture for query optimization" in Proc. of the ACM SIGMOD Intern. Conference on Management of Data, 1982
14. P.G.Selinger et al. "Access path selection in a relational database system", Proc.ACM SIGMOD Intern.Conference on Management of Data, 1979
15. E.Wong, K.Youssefi "Decomposition - a strategy for query processing", ACM TODS 1, N3 (september 1976)

MTA SZTAKI Tanulmányok 203/1988 p.159-172

ON SELECTING AND IMPLEMENTING A DATA MODEL FOR COMPUTERISED SOFTWARE MAINTENANCE AND SUPPORT

Todor Vergilov Pandeliev, B.Sc.,
Bulgarian Academy of Science - Institute of
Mathematics with Computer Centre,
1113 Sofia, Acad.G.Bonchev St., Block 8

Abstract

A brief survey of existing software engineering systems from data modelling and software description point of view is presented. Interesting trends in infological and datalogical modelling are discussed. On the basis of conclusions, drawn from the survey, concepts for a data model, suitable for software maintenance and support purposes are proposed. The latter is based on the notion of context-directed list structuring, object-oriented typing and functionality of the tools. Some implementation hints are made. The opportunity of uniting the AI power of list processing and functional programming with database management is pointed out as the major advantage of the revealed ideas.

1. Introduction

It is generally admitted that interpreting knowledge in terms of data structures, constraints and operations not only helps to represent it in a computer-processable form but also contributes to the researcher's or end-user's better understanding of real world facts.

Data modelling and database techniques have made considerable progress with business applications. This however does not hold for other areas, where organizing, storing, updating and quering large amounts of data is also inevitable, namely engineering applications, expert systems, image

processing and others ([Lock85]). On the other hand much work is being done in the field of software engineering aiming at cost reduction and overcoming the so-called "software crisis". The developments in this area are mostly design-oriented and seem to neglect maintenance and support activities especially in the usage phase of a product's life-cycle, where integral costs might be even higher than these of earlier phases due to a longer duration.

The presented paper aims at modestly contributing to filling the gaps just mentioned by outlining some features of a data model and making implementation hints, both based on conclusions drawn from a brief survey of existing systems and issues.

2. Problem specification

Three general problems exist in data modelling ([Tsich82]):

- creation or choice of a model, close to the end-user's concepts, i.e. not imposing unnatural constructs or terminology upon him - infological modelling
- choice of a conceptual model, supported or supportable by a computer system at the present (or maybe near future) state of information processing technology - datalogical modelling.
- infological-to-datalogical mapping

So far as the datalogical aspect is concerned, the alternatives available from industry, are quite limited in number. Besides it is often the case, that DBMS features, such as performance and user interface have a greater impact upon the choice than the implemented data model. Even in case of a well-grounded choice of both models, the mapping between them is still far from trivial. Solutions of the latter are attempted mainly in two directions:

- partial or specific exploitation of infological ideas, e.g. Chen's Entity-Relationship model is used mainly for schema design.
- extending datalogical models with ideas from other areas - theory of programming languages (typing, implementation hiding), artificial intelligence etc.

Other problems arise from application-specific issues. In engineering applications ([Lock85]) many entities and relationships are known beforehand with their semantic properties. This means that a general entity or relationship concept would turn out too general and universal leading to the risk of incorrect handling or poor performance. The opposite approach of including the specific entities and relationships as explicit concepts of the data model tends to make the Data manipulation language too large and complex. Another difficulty is the inhomogenity of the database contents, e.g. design data, test data, management data etc. Consistency constraints are very complex because of the complexity of the design objects themselves. Procedure plans (apply what tools in what order) should also be implemented as constraints.

Still more problems arise when software engineering applications are concerned. The data objects are very large and complex and carry meanings themselves. What is more even smaller or simpler objects (modules) do not exhibit regular properties (e.g. compare to VLSI design), so libraries tend to be very large. Modules have different representations throughout the life cycle (e.g. specification, source, object etc.) and different versions within the same representation.

In post-programming phases of the life-cycle the latter becomes still more essential. Operations and constraints for version release, configuration management and others have to be implemented. With maintenance and support functions complicated relationships exist between customers, versions and tasks ([Esk84]). The latter fall into two categories: correction (error-elimination) and perfection; each of the tasks consists of a number of activities depending on versions and customers. Some tasks (especially correcting ones) should be automatically spread over many customers who didn't request them. Tasks have a mini life-cycle - they emerge, execute and have to be archived.

3. Brief survey of some existing computerized software engineering systems from a data modelling point of view

1)AIDES

AIDES is a Hughes Aircraft product. The (infological) model is graphic in nature, representing structural and fucntional features of a target software system. The description is constructed interactively using a keyword user interface. Fig.1 contains an example (quoted from [Will80] with a minor modification), that gives a nice notion of what the model is really like. Illustrating the pragmatics of the model the following is worth mentioning: Substructures of a target software system are automatically divided into "areas of independence" and thus a testability metrics is applied.

Datalogically the choice was the Relational model, but at the time of issue of the article it was claimed not to be final. No information about further developments of AIDES is available to the author.

2)ARGUS

The system has been developed at The Boeing Computer Services. Its model incorporates two entity types: process element and data element. Process elements have an interface attribute. They fall into two categories: generic, meaning subject to further refinement and source, meaning directly implementable. The latter is aimed at achieving independence of design from implementation.

Integrity constraints are imposed over the access to a data element as well as over mutual access by more than one process element. Another constraint type is the range constraint over the data, generated by a process element.

The system was implemented using a database management system ADBMS. The choice was made following portability considerations (ADBMS is coded in FORTRAN) which is very indicative of the undue priority usually given to DBMS specific features over the data model - a fact, pointed out by Tsichritsis and Lochovsky ([Tsich82]).

3)CADES

CADES is a product of ICL (Great Britain) and was used to develop the VME/B operating system for the ICL2900. Structural aspects of the target system are taken into consideration rather than behavioural ones ([Snow81]). Two levels of representation are distinguished - structural and detailed. The software units' code is kept in separate files. At both levels a special System Description Language (SDL) is applied. Algorithms are coded in an Algol-like language, called S3.

Two entity types are available in the data model - processor and container. Defined relationship types are:kin, map (between entities of the same type) and usage (between a processor and a container).

The design process is carried out as stepwise refinement in terms of the entities and relationships mentioned above.

Datalogical aspects are interesting and quite indicative: In the early stages of the system development a hierarchical DBMS was used, replaced later by a network DBMS called IDMS with a relational outer model implemented. What is more - the schema is designed as relational as well. There is no explicit Data manipulation language - a definable extension of SDL called the Information Representation Language is used instead. Defining it plays the role of schema definition, its operators - of database operations.

4)CDL2

The basic conception of CDL is that of a single language, cross-compilation being used when the target computer differs from the host. The CDL2 language is the universal means of communication with the system. It is used as a user iterface, for design and programming of target software and so on.

The data model is hierarchical. The database is a tree with its uppermost level representing a list of owners for the purpose of access control and authorisation. The lower levels correspond exactly to the hierarchy of CDL2-language constructs: PROGRAM, MODULE, LAYER, SECTION, PROC, CALL, PAR, OBJECT. Essential database operations are the navigational ones - moving the so-called focus of interest.

Instances of modules, layers and sections are connected by means of explicit export/import interfaces. An interface from a lower-layer section to a higher-layer one is always exporting and is called abstraction. An interface within the same layer is called extension.

4. Some interesting developments

1) Version modelling ([Katz86])

The model is intended for version modelling in CAD databases. Objects are typed and fall into two basic categories: primitive and composite. A composite object is a representational hierarchy of primitive ones, the qualifier "representational" meaning that the primitive objects comprising a composite one belong to the same kind of representation (e.g. source module). Relationships in the model between different representations of objects are called equivalences and are not necessarily 1:1. Generic objects are supported by the data management system to implement the concept of versions. Each generic object has the so called version plane related to it.

Versions of an object are arranged in a tree structure with a pointer at the current version (not necessarily the latest one). Development is permissible only in the subtree corresponding to the current version node. Generating a new version of an object automatically generates instances of the equivalences it is involved in (Fig.2). A configuration is very much like a composite object, but is composed of specific versions of generic objects.

The notion of a layer is also introduced. A new layer consists of all newly created objects and the new versions of existing ones.

2) The integrated data model - IDM ([Beech83])

The model originates from the Hewlett-Packard Computer Research Centre and develops the Functional data model. It is quite representative of modern trends in data modelling and

integrates ideas from the areas of database theory, artificial intelligence and programming languages. Basic concepts of the model are: object, abstract type, relation.

Everything in the database is an object - even the database itself is an object. The approach to structuring is based upon abstraction and typing: an object is considered to be an instance of a finite number of types.

Types are objects - predefined and user-defined. Every type has a set of operations, related to it.

Operations are objects, representing functions of a finite number of parameters. Operations may be coded in one of several admissible languages.

Objects and types are linked dynamically: an object can gain and lose types. For example (cited from [Beech83]) if Jones is an instance of the types "person", "employee" and "pilot", it could later acquire the type "manager". Types are arranged in a lattice and operations are inherited according to it - e.g. it is natural to admit that operations, permissible for "person" should be available for "employee" as well.

Different objects of the same type may have different implementations.

There is a predefined type Relation, so relations are also objects. This is the only information-carrying concept in the model - there are no object attributes, properties, components or whatsoever. Types, over which a relation is defined, are called its domains. It is clear that relations of relations are possible. Predefined operations of realtion instances (called relationships) are Find, Insert, Delete and Update. Relations may be stored or derived.

5. Conclusions drawn from the survey and proposed solutions of problems

Recently it has been becoming more and more evident that the classical business-oriented approach to data modelling and DBMS design does not work properly with engineering applications ([Lock85]). The reason obviously is that it fails to meet some essential requirements of these. Its unfitness might most

generally be explained by the fact that classical databases are tailored for large amounts of uniformly structured data, that is scarcely the case with engineering applications. Besides data modelling seems to be neglected as an important factor for properly constructing a database (remember ARGUS). The Relational model is preferred in most cases (CDL2 is a remarkable exception). The motive for this is the flexibility and universality it exhibits. The price that has to be payed is the insufficiency of intrinsic constraints and poor performance. Its data structures do not correspond to the natural constructs of software engineering applications at least because these are object-oriented and objects are usually arranged hierarchically - in trees or lattices. What about a hierarchical model then? The case with CADES is quite indicative - a hierarchical DBMS has been renounced to be replaced by a relational outer model above a network one, i.e. a graph intension is avoided. To find out the probable reason for that let us try to distinguish intensional aspects (schema) from extensional ones (occurrences). With the hierarchical model the case is types arranged in a finite tree and occurrences - in a potentially unlimited set (usually ordered). What is needed in our case is almost the opposite! CDL2 illustrates the approach of tailoring the application concepts (by specially constructing a language) to the data structures of the model.

Because of objects and their structuring advanced navigation facilities are essential. In [Lock85] the possibility for tools to choose the level of granularity, at which they operate, is pointed out to be necessary. Hiding of lower level details by introducing a special type "long field" (meaning just a sequence of bytes) is proposed. Further on in this paper a better solution will be outlined.

Describing software is far from trivial. The task will be made simpler if certain aspects are clearly distinguished. At least three separate points of view to programs as data objects are possible: functional (let alone semantic issues, just environment, input and output data and so on), structural and implementational. So infologically a model, based on this

conception with uniting the three in a relational manner seems to be a good approach. There is another reason, that makes relations essential, namely the variety of different kinds of data that have to be stored, e.g. program data, customer data, task data etc. The implementational aspect causes few problems to arise - the version modelling proposed by Katz is an excellent solution.

Structuring of software is a field in which there are advanced developments and much research is still in progress. However there are a few concepts and ideas, that are generally accepted, e.g. hierarchical arrangement and module-interface. Usually the notion of process and data elements is also there. There is still considerable diversity in the ways software structuring is done and described.

The latter holds for software functioning to a much greater extent. Certain success in formally describing it has been achieved only for functional programming. In the author's mind exploitation of ideas from the area of expert systems might prove to be useful.

Datalogically the concept, uniting the (at least) three diverse aspects is the hierarchical structuring. Lists and list processing, enriched with objects and typing and relational issues like in the IDM, is obviously the most suitable candidate. Programs in one context are usually data in another. A side effect of the above choice would be the opportunity of making use of the unique feature of list-processing - representing programs (database procedures) and data in a unified manner. Of course the latter is not at all easy because of the need to take typing into consideration. Introducing an explicit "apply" facility might be necessary, meaning execution of an object with other object(s) as its input, obeying explicit constraints such as types and procedure plans.

Unlike design, where however rigid restrictions might be imposed over the structuring, interfaces, design procedures and so on (even the briefest survey is a proof), maintenance and support activities have to face a vast variety of characteristics of objects and procedures already designed. So

it is desirable that the data management makes use of the common way programs are stored - sequential files in modern hierarchical file systems. If a flexible enough structuring scheme were imposed over the software data, that would render a powerful means of building applications. We propose that the basic concept of such a scheme becomes the typed object, structured as a list. Structuring is context-driven, meaning the following: atom and list delimiters, keywords and the methods of ordering and naming are explicitly declared (also in a list-like form). We shall refer to these as "the structuring context". It is specific for each focus of interest (in other words currency indicator) and is a nice means of selectig granularity. For example a given application might be interested in tokens such as module names for cross-referencing, while another one might consider every character an atom. The structuring context concept is a contribution to data independence, so it would play the role of a specific conceptual schema. Relations are constructed in a manner, similar to that of Hypertext systems ([Del86]) with an important difference - sublists rather than byte offsets are referenced.

Operations are functions, mapping a list of objects into a single one - adding it to the database or replacing an existing one. Navigation with list-like objects is most natural and comprises selections of the next or previous element (sublist) and changing the level of nesting. Specification of an object at the selected level is content-(by atom or substring value), name- or type-driven.

7. Implementation hints

Abstract machines, oriented towards list processing are powerful and concise, e.g. Henderson's SECD machine. A major shortcoming of these is poor orientation towards direct access storage devices, inevitably needed for modern data management. In the case of software support, where mostly sequential files are used, that is not so bad. Besides the content addressability feature of the Relational model poses similar problems, which are solved by proper storage management techniques. Indexing in

terms of associative lists or lower level primitives could improve performance.

Once implemented, a list-processing abstract machine with a context-driven structuring will become the basis for the model implementation.

8. Conclusion

It is clear that the proposed ideas need further development and refinement. Future work might pose difficulties, hard to identify at the present state of the research, thus leading to the necessity of restricting or subsetting the exposed ideas and intensions in a practical system.

Some principle problems will have to be solved, e.g. the need for implementing demons and a suitable user interface.

The important thing however is, that the outlined approach yields the opportunity to unite the unique AI power of list processing and functional programming with techniques, specific for database management, which is another way to put sophisticated update (side effect) handling. Thus it becomes possible to build expert systems for computer aided software design and maintenance, backed up with data management facilities.

LIST OF FIGURES

A calls B,C and D (conditionally);
A loops from B to C;
D input is 'DATA';
B calls E or F;
D accesses table TBL;

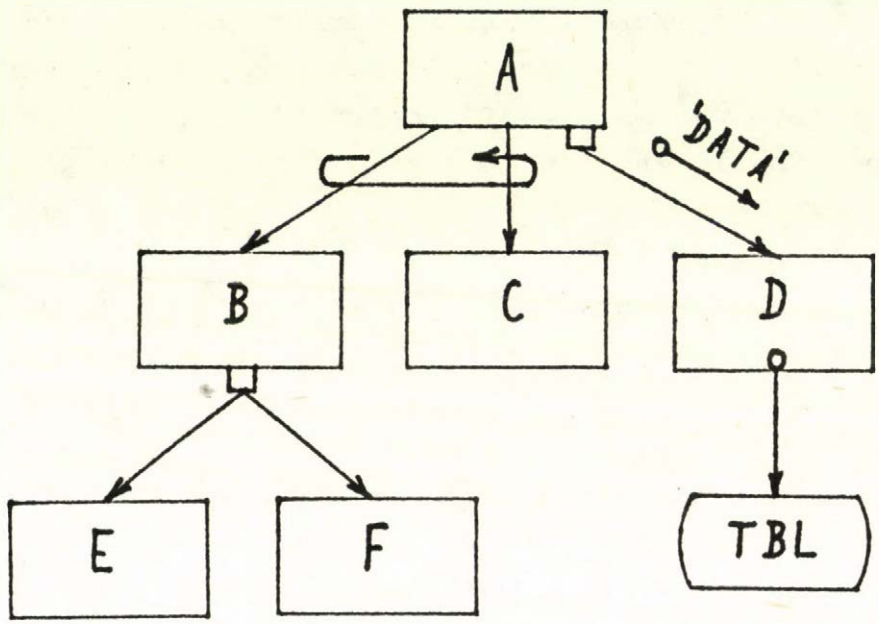


Fig 1

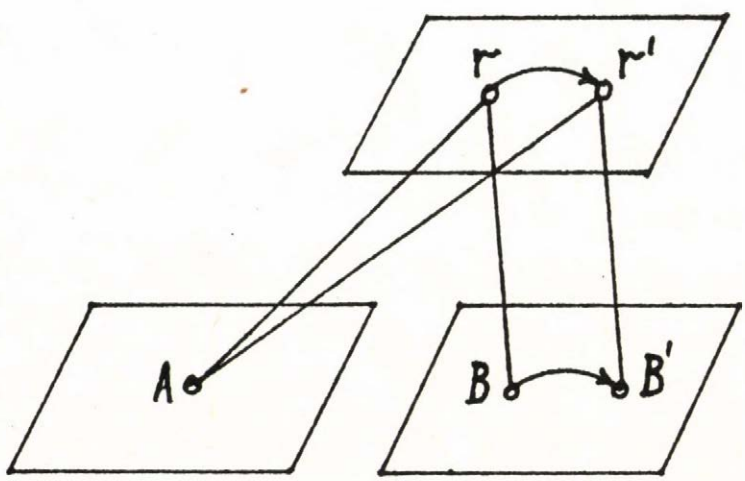


Fig 2

LIST OF REFERENCES

[Bay81] M. Bayer et al. - Software development in the CDL2-Laboratory; in: H.Huenke (ed.) - Software Engineering Environments, North Holland Publishing Co., Amsterdam, The Netherlands - 1981.

[Del86] N. Delisle, M. Schwartz - Neptune: a Hypertext System for CAD applications; in: ACM Proceedings of SIGMOD-86, International Conference on Management of Data - Washington D.C., May 1986, SIGMOD Record Vol.15, No.2, June 1986.

[Beech83] David Beech, J. S. Feldman - The Integrated Data Model - a Database Perspective; in: Ninth International Conference on Very Large Databases, Proceedings, Florence, Italy, Nov 1983.

[Esk84] А. Ескенази - Автоматизиране на една фаза от софтуерното производство; в: Автоматизирани системи за управление, кн.3 от 1984 г.

[Katz86] R. Katz et al. - Version Modelling Concepts for Computer Aided Design Databases; in: ACM Proceedings of SIGMOD-86, International Conference on Management of Data - Washington D.C., May 1986, SIGMOD Record Vol.15, No.2, June 1986.

[Lock85] P. C. Lockemann et al. - Database Requirements of Engineering Applications, an Analysis; Universitaet Karlsruhe, FZI-Pulbikation Nr.3, Juli 1985.

[Snow81] R.A. Snowdon - CADES and Software System Development; in: H.Huenke (ed.) - Software Engineering Environments , North Holland Publishing Co., Amsterdam, The Netherlands - 1981.

[Tsich82] D. Tsichritsis, F. Lochovsky - Data Models; Prentice Hall, Inc., Englewood Cliffs, 1982.

[Will80] R.R. Willis - AIDES - CAD of Software Systems II; in: H.Huenke (ed.) - Software Engineering Environments , North Holland Publishing Co., Amsterdam, The Netherlands - 1981.

MTA SZTAKI Tanulmányok 203/1988 p.173-182

NEW DIRECTIONS IN END-USER LANGUAGES RESEARCH, DEVELOPMENT AND APPLICATION

J.Penova
Institute of Mathematics with Computer Centre
Bulgarian Accademy of Sciences
BG, 1113 Sofia, Acad.G.Bontchev, bl.8

INTRODUCTION

Recently a big quantity of new languages suitable for end-user to define their own problems themselves and to find an adequate solution through Database Management Systems(DBMS) were introduced.New powerful tools intended for non-programmers or specialists were created.Using end-user languages results in a new methodology of application development.

The topics of the paper are:

- languages evolution
- basic requirements to the end-user languages
- definition and analysis of the new integrated products known as Fourth Generation Languages (4GL)
- important trends in end-user languages development
- end-user languages - new methodology of application development
- aspects of the possibly fifth generation environment

LANGUAGES EVOLUTION

The first languages - from machine language through assembly languages to the high-level programming languages were intended for programmers only.The introduction of the data independence concept 1 and the database approach led to the creation of new database languages corresponding to the three classical models.The creation of the relational languages which

have high-level operators and are non-procedural represents the last step toward the new-developed end-user languages.

On the other hand the way through which a professional and an end-user access the necessary information is rather different.The professional is familiar both with the DBMS and the access methods while an unexperienced user interacts by expecially created tools.According 2 the overall process of languages evolution is systematized in Figure 1.

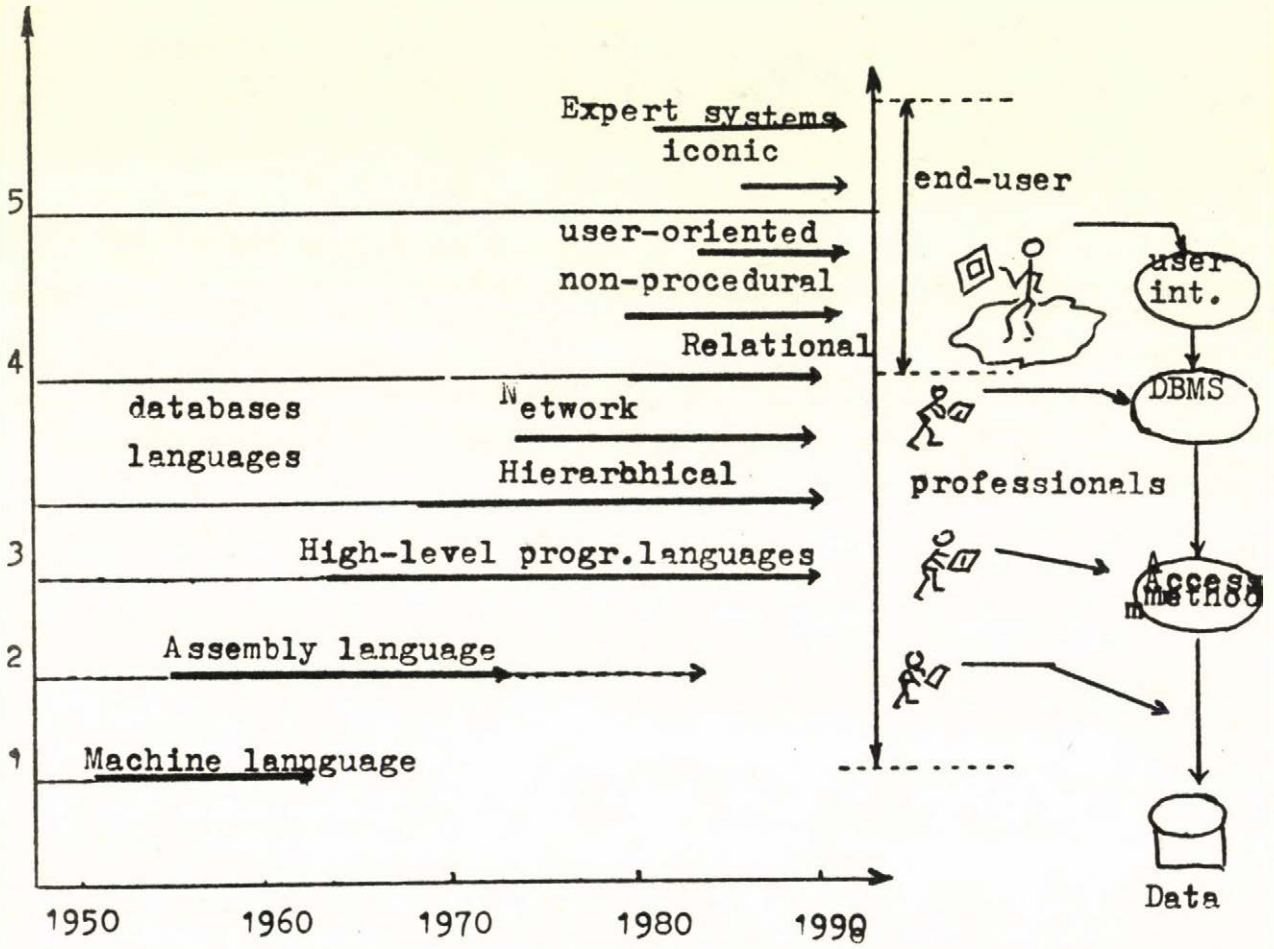


Figure 1

Recently notable forces are converging toward the development of the user interface.It consists of different hardware and software products such as:

- microcomputers
- relational DBMS

- personal software
- office automation systems
- fourth generation languages

BASIC REQUIREMENTS TO END-USER LANGUAGES

The basic requirements can be systematised as follows:

1. No data processing knowledge is necessary.
2. No information about logical or physical data organization concerns the end-user.
3. The language is "user-friendly".
4. On-line help facilities have to be available.
5. Appropriate functions(date,time,etc.) should be on hand
6. Information from other systems should be easily processed.
7. Possibilities for fast user-prototyping.

FOURTH GENERATION LANGUAGES-DEFINITION AND ANALYSIS

To give an exact definition of the 4GL is a rather difficult problem because of their complexity.Usually two different approaches are used to determine them) .One is based on several characteristics which should be presented in a given software product in order to be considered as a 4GL.The second tries to define the 4GL-s through a list of applications generator categories.

Approach A

The basic properties of this new type of languages can be systematized as follows:

1. The solution of a given problem is achieved 10 times faster than if a classical high-level language is used.
2. The language is appropriate both to end-users and professionals.
3. The user is completely unaware of the hardware details and the system software such as:access methods,screen facilities,logical and physical data organization.Each application can evolve independently,files should be easily created and modified.

4.The language includes a number of different software tools designed to facilitate the end-user.

5.Complete dictionary information about the file structure,the type of stored data,etc. is easily obtained.

6.The language is "user-friendly" i.e. it corresponds to the user's own level of knowledge.

Approach B

The applications generator categories can be sorted in ascending order according the complexity of the used means. They are:

1.Common software tools intended for personal computers(PC).They fall into two classes:

a)tools oriented toward the end-user such as:

- Lotis 1-2-3 — Lotus Dev.Corp.
- Symphony — Lotus Dev.Corp.
- dBase II,III,III Plus — Ashton-Tate
- Framework — Ashton-Tate
- Personal Decision — IBM
- Business Management — IBM

b)tools oriented toward access to a mainframe database by means of PC:

- Intellect - Artificial Intelligence
- System W - Comshare
- Focus - Information Builders
- Ramis II - Mathematica Products Corp.
- Nomad II - D&B Computing Services

2.Query languages

In our opinion query languages are only a part of the total interface between user and computer system.

Following Codd 4 the various forms of relational query languages are essentially equivalent in expressive power.Nevertheless Chandra and Harel 5 developed a complete hierarchy of query languages expressiveness:

Propositional calculus
Tableau queries
Conjunctive queries
Existencial queries
First-order queries - relationally complete
Fixpoint queries
Full first-order logic and disjunction/negation
Second-order queries
Computable queries

The levels under the relational compliteness identify special cases in query optimization.There are also optimizing translation algorithms that allow the installation of efficient navigational interfaces on relational systems and of relational query languages on network databases.However a specific DBMS product may not offer these facilities and thus introduce database dependency for the user.

The levels beyond relational completeness recently have received the attention of language designers.The set of computable queries would provide full programming language capabilities in the query languages.Thus the upper bound on selectivity will be constituted.Into the existing DBMS only a small set of efficiently computable functions has traditionally been built into the query languages.But powerful capabilities such as more general functions on the data or high-level programming languages operators are offered by the other applications generator categories.

3.Report generators.

By means of them the user specifies only the format of the required report and the corresponding procedure is automatically executed.

4.Graphical tools.

Using them different diagrams,graphics can be displayed on the screen.For mainframes one of the best products is TELL-A-GRAPH - Integrated Software System.

5.Decision support and financial modeling tools.

They are used for data analysis and different models creating.Answers of "what-if" questions depending on the

actual stored data can be also obtained.They fall into two classes:

a)spreadsheets for PC
- Lotus 1-2-3
- Framework
- Multiplan
- Visicale

b)decision support systems for mainframes
- System W
- Express

6.Applications generators.

These sortware products ensure the creation of complete applications.They can be subdivided into:

a)end-user oriented - Focus,Ramis II,Nomad
b)professionals oriented - Ideal - Applied Data Research

Obviously a realy good 4GL would contain all of the listed above categories.However only some of them are included consider their complexity.Practically the creation of a 4GL is a very difficult process because of their extremely high requirements.In our opinion it will be very important to investigate such kind of languages for microcomputers and to find out the query optimisation possibilities.

SOME IMPORTANT TRENDS IN 4GL DEVELOPMENT

The 4GL represent powerful and integrated software products with enought rich and various possibilities.Nevertheless there are some directions in which they have to be improved taking into account some modern hardware tools,new approaches to data processing and the increasing user interest.We mention the following important trends:

1.Support of PC versions.

Recently an expanded usage of PC-s for data processing and design applications is noticed.The PC-s can work as a terminals to look up the current information or to update the stored data in the mainframe computer.They also can be all connected together on a local area network.That is why

a PC version of a 4GL is extremely desired.It will permit the generated applications to be executed both on PC-s or mainframes interchangeably.Unfortunately up to now only few of the created 4GL-s - for example Focus,Ramis II,Nomad,Mantis - have a PC version.Some difficult problems to be solved arrise such as:

- data transparency
- fast interaction with the so-called "host" computer
- fast user interface with the distributed database system

2.Further integration and automation.

The future trend in this area is oriented toward the usage of functional diagrams and the determination of data representation standards.Further development of the relational database systems is expected.New tools by integrating a 4GL with application software packages will be obtained.

3.Further improvement of end-user tools.

Most 4GL-s require a rigid command syntax.An exeption of this rule is Tell-a-graph which has very flexible set of commands.From this point of view few of the 4GL-s offer convenient end-user interfaces.Among them Focus and Ramis II can be mentioned because of their interface variety.For example Focus has well-designed menu-driven interface called Tabletalk. It helps the unexperienced user and simultaneously possesses a strong command syntax.Obviously the usage of 4GL-s is rendered difficult on account of the computerised syntax.For this reason a future development of new interface techniques more convenient for user is expected.

END-USER LANGUAGES - NEW METHODOLOGY OF APPLICATION DEVELOPMENT

Using the 4GL-s leads to a new way of application development.The end-user interacts directly with the system dinamically introducing eventual modifications and receives the results very quickly.The program development time might be reduced by 50% to 80%.The productivity shows a notable

improvement because of the end-user prototyping.It is well known that by means of a classical language the response of the user comes after several months of development.

Recently some integrated software packages for microcomputers which possess the 4GL characteristics were introduced Among them the Ashton-Tate products - dBase III and the last version - dBase III Plus can be mentioned.They offer:

- relational database processing
- multiple databases handling
- various data fields types
- access to DOS commands
- mathematical capabilities
- new programming powerful techniques
- LAN possibilities (dBase III Plus)

These products are expecially convenient for the creation of the following applications:

- information-retrieval systems
- facts management systems
- inventory and account receivable systems
- statistical and financial packages

Obviously the creation of different applications in various fields is possible.We used these products expecially for information-retrieval systems development.As a concrete example we developed a system which aids the organization and holding of a scientific meeting(symposium,seminar,conference). Its user is menu-driven in different directions such as: data input concerning the participants and the meeting information, generation of lists and total inqueries,database administrator utilities.The dBase III structure ensured the realization of the necessary functions.This example and some analogous applications will help our query optimization problem research.

FIFTH GENERATION ENVIRONMENT - POSSIBLE ASPECTS

The fifth generation will contain new powerful tool coming from Artificial Intelligence Studies.Among them knowledge bases and expert systems have to be considered as

possible aspects of the future end-user environment.

An expert system 7 is an information system that can pose and answer questions relating to information stored in its knowledge base.It mimics the deductive or inductive reasoning of a human expert and the adequate decision is automatically extracted from the data describtions by a user-invisible inference procedures.Thus,an expert system is a problem-solving software product that solves substantial problems generally conceded as being difficult and requiring notable expertise.

The future programming environment will be also enhanced with the icon-based languages in which visual symbols can be manipulated to formulate different queries.The entities and relationships in the database are represented by specific geometrical shapes.Multiwindow browsing improves the interaction between the end-user and the stored data.First example of such a language is Query-by-Example 8 ,based on the relational model.It makes relations directly visible as tables to be manipulated on the screen.Several its extensions are proposed such as Query-by-pictorial-Example from Chang and Fu 9 . The appropriate icons to represent objects,the use of color and highliting greatly influence the success of this new type languages.

REFERENCES

1.Date J. "An introduction to database systems",1986 (IV-th) edition, Addison&Wesley

2.Frasson C. "The evolution of end-user languages",International symposium on New Directions in Computing, august 12-14, 1985, Norway

3.Mimno P. "Fourth generation languages - power to user", Computerworld,8.IV.1985

4.Codd E. "Relational completeness of database sublanguages", In Data Base Systems,R.Rustin,Ed. Prentice-Hall,1972

5.Chandra A.,Harel D. "Structure and complexity of relational queries",J.Comput.Syst.Sci. 25,2,1982

6.Jarke M.,Vassiliou Y., "A framework for choosing a database query language",Computing Survey 1985,Vol.17,3
7.Negoita V.,"Expert systems and fuzzy systems",1985
8.Zloof M. "Query-by-example:a database language",IBM System Journal,16,4,1977
9.Chang N.,Fu K. "Query by pictorial example", IEEE Transaction of software engineering, vol.SE.06,1980
10.Simpson A. "Advanced techniques in dBase III",1986
11.dBase III(TM) "dBase III Reference guide", Ashton-Tate,1984
12.Lima T. "Mastering dBase in less than a day", 1986
13.Williams A. "What,if...?;a user's guide to spreadsheets on IBM PC", 1984
14.Seybold P. "Integrated desk-top environment",1985, McGraw-Hill Book Company

MTA SZTAKI Tanulmányok p.183-188

Applications software for PC LANs

Gábor Remzső
Technical University of Budapest

Introduction

Most network software used today is single-user. Depending on the specific operating system and the application, this situation can cause considerable grief to the user.

Many new users who run applications on a network discover some surprises. Often the first thing a new network user wants to do is put a favourite application program on the hard disk and then share the program. Here the implicit assumption is that the software works on a floppy, so it ought to work on a hard disk. Most of the time, however, this assumption is wrong because the program is usually on a copy protected diskette. **Equally wrong, but much more dangerous,** is the assumption that **single-user data base management software can be used in the shared environment of a local area network.**

New multiuser software has opened up a wide range of software functions not possible in the single-user world. In this publication we'll examine the kinds of multiuser software now available.

Disk sharing

Today probably more than 50 percent of all local area networks are used for simple disk sharing. Programs calls to the floppy drive are remapped out on the network to the shared disk. This process lets you put on the network your current floppy software and use it with a hard disk. On the hard disk you can store either a program or data. Because network programmers have made the network look like a single-user environment, less emphasis has been placed on the information sharing or communications capability of the network.

By the same token, people are using networks without understanding network problems. Users assume that when they use a network, sharing of communications and information is implicit. It isn't! The confusion is due partly to the hardware companies' telling us that we can share files and information. What companies really provide are only **tools for sharing. Good multiuser applications software is needed to put these tools to work.**

In early multiuser systems, users often lost their data. The problem was attributed to the computer, the hard disk, or the software. Actually, all these elements were functioning properly. The problem was simple. Programs that had begun as single-user software put on the network in a multiuser environment. Then people started to change data simultaneously, **but the software wasn't written to handle simultaneous use!** A single-user software program thinks it's using a private hard disk, and the network operating system may support this assumption. The user is therefore responsible for understanding how the system works and what protections are needed.

Read-Only Application Software vs DBMS

Not all software needs to have multiuser features to perform adequately on a network. Some applications are inherently single-user. They' ll work properly whether they are run on a single PC or a network.

Word processing, graphics, spreadsheets software are single-user applications. (E.g. there is a little reason why you should let someone else modify a spreadsheet while you are modifying it.)

Right now, the problems of concurrent access relate primarily to data bases and data dase management systems. (DBMS) . Most companies or officies have only one data base for a category of information. In fact, only one data base should exist so that everyone updates and uses the same information.

A DBMS is applications software that lets the user manipulate and perform several useful operations with the data base. As a common source, the data base should be fully to everyone who needs to use it.

The networked DBMS is responsible for making the data base available simultaneously to multiple users, while protecting it from the problems that result from multiple access. Single-user DBMS software can be run on a network but may invite disaster.

Requirements of a PC LAN DBMS

- **Handle concurrent Access Contention**
- **Insure Data Privacy**
- **Provide Multi-User Security**
- **Allow Performance/Capacity Growth.**

A Multiuser Approach

The operating system can protect single-user applications in several ways. If one user opens a file and a second also attempts to open it, the network can automatically deny file access to the second user. In a sense, this approach is multiuser. When applications require that several people

open and access the same file, the approach is inadequate. In the example that follows, consider what happens on a network if the DBMS has not been written to handle multiuser access.

Two users each load a copy of the data base manager software. One common characteristic of the DBMS is to make as few writes as possible because disk access slows the system down. When loaded, the DBMS is therefore held in memory. If user A wants to retrieve a record, the information on where it is and how to retrieve it is found in the memory of **user A's** PC, not in the disk server. Since the two PCs are not sharing each other's memory, they do not share the data, but only copies of the data. As long as the data on the disk remains unchanged, no problems occurs. But let's suppose that **user A** deletes a record from a data base that **user B** is also using. **User B** should somehow be informed of the change. Otherwise, both users have the potential of destroying one another's work.

A network operating system can provide locking down to the **record level.** But locking the record, deleting it, and unlocking it will not solve the problem outlined here. The information found in the memory of one PC dose not match the information in the memory of the other PC. **User B** somehow be told to reread the data, and **user A** must be told to write the update so that when **user B** reads it, the data will be accurate. A single-user DBMS does not perform this task because it isn't significant to a single-user environment.

The only solution of this problem is to use a multiuser approach so that the multiuser software writes the file out to disk and updates it properly. With this procedure **user B** knows when a change has been made and when to reread the data. In this way, multiuser DBMS systems, such as DATAFLEX, DATASTORE, dBASEIII PLUS, manage multiple write to disk. When a record is marked, the multiuser DBMS informs the file on the disk. **User B** looks at the file on the disk and sees that it has been modified. This procedure lets **user B** know that the information and index are invalid and that the file must be reread.

When **user A** opens up a data file, deletes or modifies a record, a bit is set on the file on disk. This bit indicates that the file has been modified. Before **user B** does anything with the data, the application checks the bit and sees that the file has been modified. The program rereads the data out of the index file. Then the program retrieves the data and performs an update on the file in the memory of the PC. These two users appear to be sharing memory, but they are really sharing only the disk and synchronizing its use.

Categories of Network Software

Three categories of network software are available. They will probably change over time, but they are useful now in charting progress in software for local area networks. The software categories are the following:

- **Unnetworked**
- **Networked**
- **Multi-accessed networked.**

Software may be **unnetworked** because it abuses the operating system through some unproper scheme. Or the network may be poorly designed and therefore may not offer transparency. Today most PC software written according to the rules and uses operating systems properly and most networks adequately provide transparency. Generally, the reason why software is unnetworked is that is either hard coded for floppy diskettes or copy protected and thus is unable to reside on a hard disk.

The second category, transparently networked software, contains single-user software that can be run in a network environment. To run single-user software, the network creates the illusion of a single-user system. The software expects to "see" local resources (disks and printers), so that network simulates them with virtual devices. To fit into the networked category, software should be network-deliverable. This means that software can reside on a mass storage device and be delivered to users across the network.

The third category is the multiuser or multi-accessed network software. This applications software is aware of the multiuser environment, unlike software in the previous categories. In this software a number of techniques are used to allow data to be shared in an orderly way.

Upgrading Single-User DBMS Software

Many existing DBMS vendors will be upgrading their products to function in a multiuser environment. Some hazards and deficiences accompany any simple upgrade, however. The first thing most DBMS vendors will do is to add concurrent access in the form of **lock** and **unlock** command for the data base. Whether file lockong or record locking, or both, the commands must be explicitly given by the user. Lock and unlock commands are usually satisfactory if only two people are using the network. But three or more mulriple users can quickly be confronted with a **dead-lock** , in which case each user has possession of a file needed by another user. Here is an occasion where people need the same files but are getting in each other's way, thus effectively deadlocking the data base. And with additional users, the problem gets even more complicated.

If software offers commands only for lock and unlock, problems can crop up. As a user, you must understand the consequences of concurrent access and work out some method to avoid conflict. In most cases you will require a noncomputer, **person-to-person** scheduling of accesses, sometimes referred as a "verbal semaphore." The DBMS software will provide enough sophistication so that the user doesn't have to be an expert in solving any of these problems.

Efficient Use of the Network

Certain software pograms are better suited to networking than comparable programs in the same application. For instance, a program that runs mostly in memory is more efficient than one that accesses the disk frequently. Even in a single-user environment the number of disk accesses will slow down response time. In a network environment, disk-access requests must contend with requests of other users,so that a high number of requests may reduce performance to an unacceptable level.

Older software programs are often guilty of a wasteful number of access requests, at least from a networking perspective. These programs were written at a time when little amount of RAM was considered standard. In order to deliver functionality but stay within the memory constraints, programmers left much of the applications on disk, to be picked up as needed.

Programs may be written under the assumption that they are supported by a very large and fast, dedicated hard disk. Instead of using disk time as a shared and limited resource, these programs waste disk time with random writes. Graphics applications and some data base managers are guilty of this wastefulness. Programs that use disk requests wastefully will look much worse on a network, compared to programs that run out of local memory.

References

1. **J. Martin,** Principles of data-base Management
Prentice-Hall, Inc., Englewood cliffs, New Jersey 1976.

2. **Стогний А.А., Глазунов Н.М.** Проблемы интеграции в базах данных
MTA SZTAKI Tanulmányok 194 1986.

3. PC LANS vs. Multi-user Systems
Architecture Technology Corporation , Minneapolis, Minnesota , 1986.

4. **J.S. Haughdahl,** Inside the Token-ring
North-Holland, 1987.

5. **T. Remzső,** Office automation and data processing
In: MTA SZTAKI Tanulmányok 194 1986.

6. **G. Remzső,** Computer-aided database management system in Hernád Március 15. Agricultural Cooperative - A Case Study
In: MTA SZTAKI Tanulmányok 194 1986.

MTA SZTAKI Tanulmányok 203/1988 p.189-192

Sequential methods for monitoring side effects in a pharmacological study

M. Csukás[x], A. Krámli[xx] and J. Soltész[xx]
x Hungarian Institute of Cardiology
xx Computer an Automation Institute, Hungarian Academy of Sciences

A pharmacological study - comparing two preparates and a placebo - is being carried out for the Richter Gedeon Pharmaceutical Company. The study - according to the international standards - is based on fixed sample statistical methods. However the side effects are monitored by using sequential procedures (for mathematical backgrounds cf. e.g. "Restricted Sequential Procedures", Armitage, P., Biometrics, 16, 9-26). The sample size is 2500 patients, and about 50 side effects are continuously being monitored. The structure of data is described in the talk "Mikrocomputer-based special medical information system" (Kerékfy, P., Kiss, A., Ratkó I., Ruda, M.). Here we shall point out only one peculiarity of the monitoring problem.

For each side effect two files were constructed: the first one contains records on the patients suffering from the given side effect and taking the first preparate or the placebo while the second file contains records on the patients suffering from the given side effect and taking the second preparate or the placebo. The two types of files were processed separately. Further on - for the sake of simplicity - our considerations refer to one of the above constructed files.

Research partially supported by Hungarian National Foundation for Scientific Research, Grant No 1066

The records of the files are sorted on ascending key, where key is the time interval from the beginning of the treatment until the first occurrence of the given side effect. This arrangement is the appropriate one for the sequential procedure.

The goal of the sequential procedure is to determine whether one of the two preparates or the placebo causes a given side effect with greater probability. The probability in question is unknown, and its value is irrevelant, because the sequential procedure omits the "indifferent cases" i.e. the patients who do not suffer from the given side effect.

We assign to the ith record (on the sorted file) the value r(i) where

$$r(i) = \begin{cases} +1 & \text{if the ith patient takes the preparate} \\ -1 & \text{if the ith patient takes the placebo} \end{cases}$$

According to Wald's method, if the values r(i) form a random sequence and the probabilities of the occurence of the given side effect are π_1 and π_2 when the patient takes preparate and placebo, respectively,

then $w(n) = \sum_{i=1}^{n} r(i)$ is a random walk which steps +1 with probability

$$\vartheta = \frac{\pi_1}{\pi_1 + \pi_2}$$

and -1 with probability $1-\vartheta$.

Testing the 0-hypothesis $\pi_1 = \pi_2$ is equivalent to testing the 0-hypothesis $\vartheta = 1/2$. In the practice instead of testing $\vartheta = 1/2$ we always compare two simple hypotheses $\vartheta = \vartheta_1 > 1/2$ and $\vartheta = \vartheta_2 = 1 - \vartheta_1 < 1/2$. The ratio $\vartheta_1 / \vartheta_2$ can be given on the basis of medical consideration.

For given ϑ_1 and ϑ_2 the general scheme of the sequential procedure looks like as follows: for given probabilities of the errors of first and second kind (α and β) the coefficients a and b can be computed:

$$a = \frac{2 \log \{(1-\beta) / \alpha\}}{\log \{\vartheta_1 / (1-\vartheta_1)\}}$$

$$b = \frac{2 \log \left\{\frac{1}{2} \vartheta_1^{-\frac{1}{2}} (1-\vartheta_1)^{-\frac{1}{2}}\right\}}{\log \{\vartheta_1 / (1-\vartheta_1)\}}$$

We accept the hypothesis ϑ_1 if

$$\min_{w(n) > a+bn} n \quad < \quad \min_{w(n) < -a-bn} n$$

i.e. the random walk w(n) hits the upper boundary a+bn earlier than the lower boundary -a-bn.
Figure 1 illustrates the behaviour of w(n) for one of the side effects and for the first preparate

For a reasonable choice of the parameters e.g. $\vartheta_1 = 0.7$, $\alpha = 0.025$ and $\beta = 0.05$ the coefficients of the boundary lines are a=8.59 and b=0.2058. So in our example the sample size is not large enough to accept any of the two hypotheses.

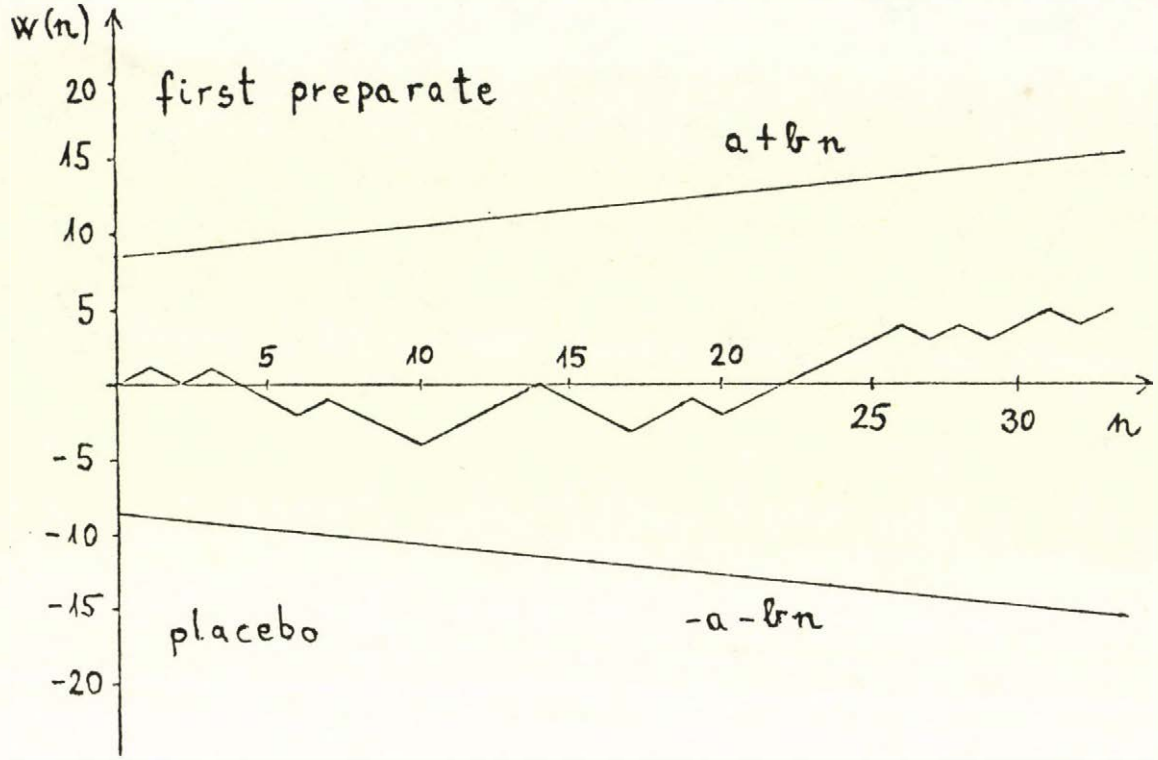


Figure 1

A technical remark: the treatment begins for different patients at different times, therefore all "side effects files" are to be rearranged when a next step in the sequential analysis is needed.

MTA SZTAKI Tanulmányok 203/1988 p.193-198

A stepwise non-parametric decision procedure for diagnosing

Margit Csukás,
National Institute of Cardiology, Budapest
A. Krámli and J. Soltész,
Computer and automation Institute of the
Hungarian Academy of Sciences, Budapest

1. INTRODUCTION

The basic problem of the statistical discrimination can be formulated as follows: there are several populations and a sample of individuals from each. Upon measurements from these individuals a certain rule should then be set up which enables us to put a new individual by its measurement values into the population to which it belongs. When the results of these measurements - the so called explanatory variables - can be regarded as random vectors having joint normal distribution with common covariance matrix and different mean vectors for different populations then the well-known methods of linear discriminant analysis can be used. Usually these methods are applied in any cases when the explanatory variables are continuous and the hypothesis of the multivariate normality cannot be refused. (This does not means that the explanatory variables have in fact joint normal distribution!)

In order to avoid the problem caused by the failure of the multivariate normality and to involve discrete explanatory variables in the discriminant analysis one should turn to the nonparametric decision procedures.

In this paper we propose a stepwise nonparametric decision procedure which was constructed for special medical diagnostical purposes. After the description of the algorithm we give a detailed discussion of the results obtained for the medical problem which was the origin of our investigations.

2. THE DECISION ALGORITHM

First we introduce some notations. The number of populations is two,

Research partially supported by Hungarian National Foundation for Scientific Research, Grant No 1066

they are denoted by CLASS1 and CLASS2, respectively. For every explanatory variable there exists a natural ordering $>$. $A :=$ $:= \{ 1, \ldots, n \}$ is the set of individuals belonging to the sample ; $x_1, \ldots, x_k$ are the explanatory variables; the random vectors $\underline{x}^{(i)} = (x_1^{(i)}, \ldots, x_k^{(i)})$ are the explanatory variables of the ith individual so $\{ \underline{x}^{(i)}, i \in A \}$ is the statistical sample for the construction of the decision procedure; finally A_1 will denote the result of the decision procedure: the set of individuals classified into CLASS1. Remember that for A the true classification is known. The number of "false negatives" is m_1 ($m_1 = \#(\text{CLASS1} \cap (A \setminus A_1))$ - the symbol $\#B$ denotes the cardinality of B), while m_2 denotes the number of "false positives" ($m_2 = \#(\text{CLASS2} \cap A_1)$). By defintion the expected error rates α and β are the expectations of the ratios $m_1/\#\text{CLASS1}$ and $m_2/\#\text{CLASS2}$, respectively.

The linear discriminant analysis minimizes the posterior expectation of m_1+m_2 (or that of some linear combination of m_1 and m_2), cf. e.g. R (1965) Section 8e. The objective of our nonparametric procedure is to minimize a monotonous function of m_1 and m_2 namely the so called Fisher's exact probability: $F(A_1) := \text{Prob}(\#(B \cap \text{CLASS2}) < \#(A_1 \cap \cap \text{CLASS2}))$ where the probability is computed assuming that a subsample B of size $\#A_1$ is drawn at random from the whole sample A ; i. e. all subsamples of size $\#A_1$ are equally likely . This probability depends only on $n_1 := \#(A \cap \text{CLASS1})$, $n_2 := \#(A \cap \text{CLASS2})$ $l_1 := \#(A_1 \cap \text{CLASS1})$ and $l_2 := \#(A_1 \cap \text{CLASS2})$ and it will be denoted by $P(n_1, n2, l_1, l_2) =$

$$= \sum_{\substack{0 \le r \le l_2 \\ r \ge l_1 + l_2 - n_1}} [\, n_2! * n_1! * (l_1+l_2)! * (n_1+n_2-l_1-l_2)! \,] / [\, (n1+n2)! * r! * (n_2-r)! * (l_1+l_2-r)! * (n_1-l_1-l_2+r)! \,]$$

We impose the following constraint on the decision procedure. The subsample A_1 should be formulated via a logical expression given by a disjunctive normal form. This means that there exist a sequence $\{x_1, .., x_s\}$ of explanatory variables, a sequence $\{c_1, \ldots, c_s\}$ of threshold values and a sequence of symbols $\{\perp_1, \ldots, \perp_s\}$ ($\perp_j$ means $\le$ or $>$) such that

(*) $i \in A_1$ if and only if $[(x_1 \perp_1 c_1)\&(\quad)\ldots]$ or$[\ldots]$or... or$[\ldots\&(x_s \perp_s c_s)]$.

At present it is not known whether there exists an effective (polynomial in n) algorithm to find a decision procedure of the form (*) which minimizes $F(A_1)$. Therefore we have used the following, intuitive, short-sighted algorithm.

The algorithm contains two parameters P_{stop} and s which should be fitted to the given problem.

Step 1. Look for an index j, a threshold value c and a symbol $\perp$ such that

$P(\#(A \cap CLASS1), \#(A \cap CLASS2), \#(B \cap xCLASS1), \#(B \cap CLASS2))$ is minimal, where $B := \{i \ ; \ x_j^{(i)} \perp c\}$. If $P_1 := P(\#(B \cap CLASS1),$

$\#(B \cap CLASS2), \#(B \cap CLASS1), 0) > P_{stop}$ then the individuals $i \in B$ will be labelled with F ("finally classified")

($P_1 > P_{stop}$ intuitively means that even the best possible partition of B can be occured by chance).

If $P_1 \leq P_{stop}$, then the individuals $i \in B$ will be labelled with 1 ("classified into CLASS1"). The individuals $i \in B$ will be labelled with 2 ("classified into CLASS2").

Step 2. Let B_1 and B_2 be the set of individuals labelled with 1 and 2, respectively. If B_1 is non-empty then replace A by B_1, else replace A by B_2. Continue the algorithm from **step 1**, while the number of executions of **step 1** does not exceed s.

Step 3. The individuals labelled with F or 1 will constitute the resulting subsample A_1.

In the next paragraph the construction of the algorithm will be illustrated by a graph.

3. DISCUSSION: A MEDICAL EXAMPLE.

First let us say some words about the data. In 9 medical centres of different countries 297 patients - suffering from a special lung disease (COLD) were investigated. 33 relevant characteristics of them were measured by noninvasive methods (such as blood pressures, ECG waves), so $n=297$, $k=33$. In addition the so called PAP value was determined for each patient by an invasive method. CLASS1 consists of patients having pathologically high PAP value.

First we carried out the standard methods of linear statistical analysis using 18 variables (both discrete and continuous) from the 33 ones; 4 variables entered into the discriminant analysis which resulted in the following classification matrix

	A_1	$A \setminus A_1$
CLASS1	43	20
CLASS2	37	180

So $m_1=20$ $m_2=37$, the estimates for the misclassification rates are $\alpha_{est}=20/63$ $\beta_{est}=37/217$. Notice, that the classification procedure has omitted 17 cases, because the values of variables entered into the procedure were missing.
After several trials we have chosen $P_{stop}=0.001$ and $s=11$ for the stepwise decision procedure. On Figure 1 one can follow the construction of the decision procedure. The corresponding decision rule can be written as follows:

The ith patient is classified into CLASS1 (i. e. $i \in A_1$) if and only if

(**) [(PO2R≤50)&(HRATE>85)]or[(NEGTW1>0)&(NEWPCV>44)&
&(RWAVEAVR>1)]or[(NEWHGB>159)&(PHR≤377)]or[(NEWVC≤224)&
&(SWV5>7)] or[(RVWDIAS>7)&(RVDC>30)]

Remark. If a variable $x_j^{(i)}$ involved in the algorithm had missing value at a given stage, then the ith individual was labelled with 2.

We have got the following classification matrix:

	A_1	$A \setminus A_1$
CLASS1	55	11
CLASS2	20	211

Because of the large number of variables involved in the decision procedure the error rates α and β are underestimated by 11/66

and 20/231. In order to obtain more realistic estimates for α and β one should apply different resampling procedures (cf. E (1979)). The simplest and most natural one is the so called "delete one" cross-validation: each individual i is classified into one of the groups according to the decision procedure determined from all the data except $\underline{x}^{(i)}$. The classification matrix obtained by this way looks like

	A_1	$A \setminus A_1$
CLASS1	45	21
CLASS2	28	203

The estimates for the error rates are $\alpha_{est}=21/66$, $\beta_{est}=28/231$.

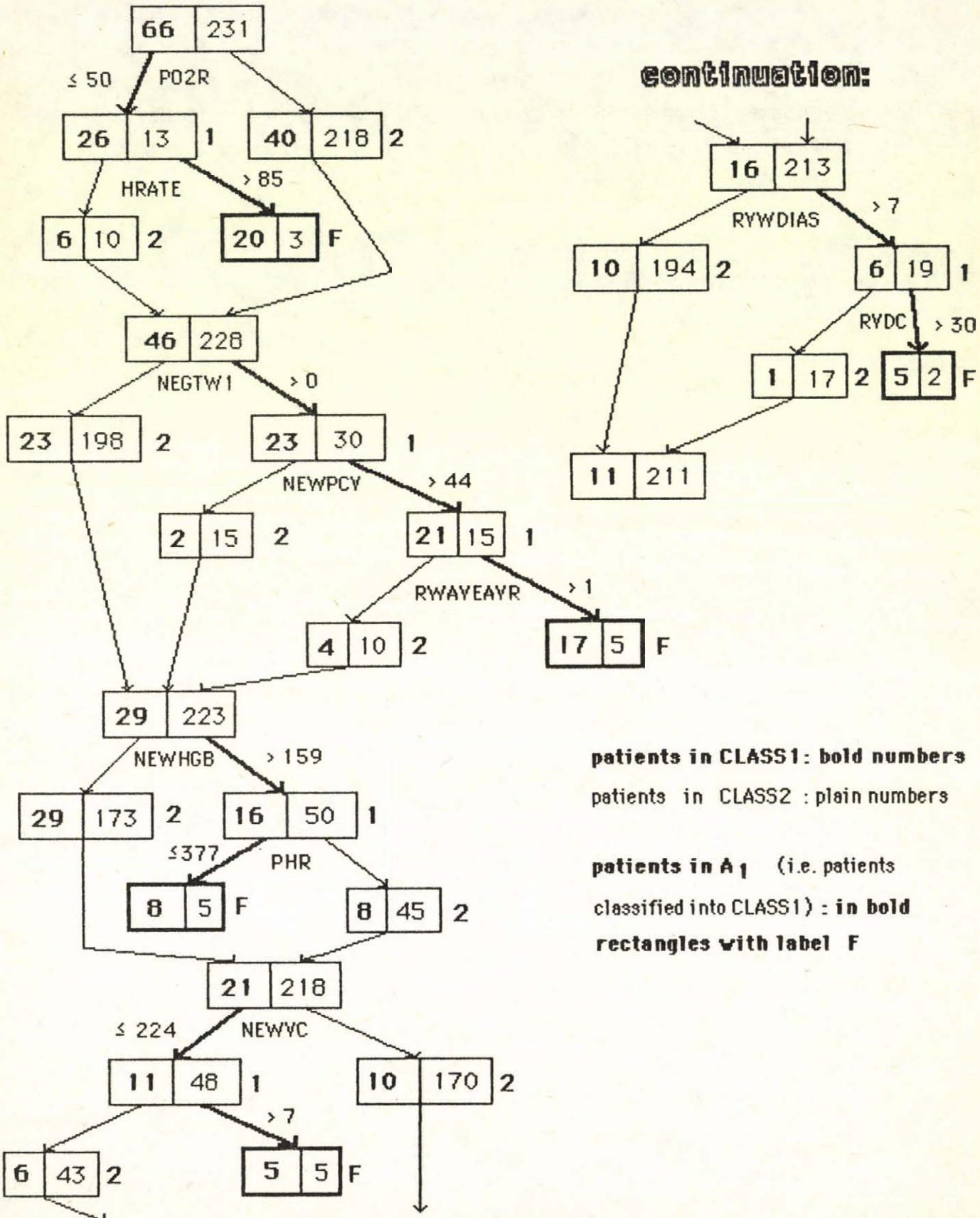
Remark. Another possibility to improve the estimates of the error rates α and β is the so called bootstrap method (cf. E (1979)). It can be summarized as follows:
select a random "bootstrap" sample with replacement from the original one; let the size of the bootstrap sample is n (in our case n=297), too. Construct a decision graph on the basis of the bootstrap sample and using this graph compute the error rates α_{or} and β_{or} for the original sample and the error rates α_{boot} and β_{boot} for the bootstrap sample, respectively. The improved estimates are $\alpha_{im}=E(\alpha_{or}-\alpha_{boot})$ and $\beta_{im}=E(\beta_{or}-\beta_{boot})$; the expectations can be computed from a sample obtained by the independent repetition of the resampling procedure . In our case the results obtained after 20 repetitions were in good agreement with those obtained by the cross-validation method.

REFERENCES:

Efron, B. (1979) Bootstrap methods: another look at the jackknife, The Annals of Statistics, V. 7, N° 1, pp.1-26.

Rao,C. R. (1965) Linear statistical inference and its applications, J. Wiley & Sons, New York - London - Sydney

The decision graph



patients in CLASS1: bold numbers

patients in CLASS2 : plain numbers

patients in A_1 (i.e. patients classified into CLASS1) : **in bold rectangles with label F**

Figure 1.

MTA SZTAKI Tanulmányok 203/1988 p.199-207

OF THE ARABLE CROPS

Gy. Biczôk, B. Lásztity
Research Inst. for Soil Sci. and Agric. Chem., Hung. Acad. Sci.
A. Bákássy, A. Krámli, M. Ruda
Computer and Automation Inst., Hungarian Academy of Sciences

Abstract

Modelling problems of nutrient element cycles in the arable soil-plant systems are investigated in this paper. It is regarded as the primary task to synthetize the resultant of the biological, chemical, pedological processes of high complexity in a unified model equation, considering the variability of the plant species, the climatic conditions and the genetic soil types. The most suitable instrument is the plant itself for tracing the complex processes and their overall effects, since the plant exactly includes the aforementioned processes by its own development or growth at a given place under given circumstances. A representative sample has to involve relatively few time-point during the vegetation period because of the remarkable cost of the soil and plant analyses. The sampling in only main phenological stages can not form a proper data base to fit sophisticated growth models, moreover beyond the accumulation the decomposition (so-called reflux) must also be described as the significant part of the soil-plant nutrient cycle. Cosequently the plant genetics limitation is applied to reduce the degrees of freedom, to avoid some degenerated solutions and thus the model identification may be performed more quickly and reliably.

Keywords: compartment model, growth theory, nutrient uptake, nutrient accumulation, phenodynamics, soil-plant system.

Research partially supported by Hungarian National Foundation for Scientific Research, Grant No 1066

Introduction

Nowadays agrochemical and computational tools has great importance for fertilizer recommendation purposes as well as for the environmental sciences. The main goal is to work out such an optimal fertilization strategy, which minimizes the damaging influence to the environment. To performe this it is sufficient to know the nutrient uptake dynamics of soil-plant systems. Since the plant and soil analyses are very expensive procedures the authors have developed a model [1] which requires a few data only and which can give an overall description of the practical events in the soil-plant system, while its accuracy is sufficient for developing suitable technologies for plant cultivation in a farm.

Our phenomenological model is based on values measured at the main phenophases in the vegetation period. Our aim is to recognize the dynamics of the accumulation. This is the cause why the term of phenodynamics is used. The system analysis of the model has been published in [2]. The papers [3] and [5] contain a more detailed discussion. The model was identified by carrying out systematic observations over several years in all important agro-ecological regions in Hungary. The model identification was processed for 12 nutrient elements and the dry matter, for the 10 most important arable crop species using about 6000 series of measurements in different soil and climatic circumstances. Several authors published case studies using our model (see e.g. [4], [7-9]).

The results were compared with the well-known growth models: those of Gompertz, Mitscherlich, Janoschek and Richards [10].

1. The model description

Our model describes the nutrient accumulation dynamics of soil-plant systems. The overall process of plant mineral nutrient element accumulation (and the dry matter accumulation) is controlled by the available quantities of the nutrient elements in soil and the accumulated quantities of the nutrient elements in plant. The simple proportionality between the quantity uptaken

by plant and that in soil, respectively leads to a Riccati-type differential equation having logistic time dependence function as a solution. Supposing a similar dynamics of the nutrient elements for the losses of plant (reflux) a so-called two-compartment model can be fitted to the data of the nutrient element and dry matter accumulation of different arable crops under numerous environmental conditions. The paper [5], which contains a detailed analysis of our model, conclude that the resultant of the accumulation process and the reflux looks like:

$$U = \frac{A}{1 + e^{-b(t-t_g)}} + \frac{R}{1 + e^{-s(t-t_s)}}$$

where A (available nutrient element quantity) is the upper limit of the accumulation, b (nutrient element buffer capacity) is the deceleration of the nutrient element supplying process in the generative period, the parameter t_g represents the time point of the maximum accumulation rate (that is the inflexion point which means the begining of the generative plant growth phase), R is the total reflux (the total loss of plant), s is the buffer capacity of plant which decelerates the senescent lossing and t_s is the inflexion point in the reflux caused by the senescence.

Now we demonstrate the behaviour of the soil-plant systems described by such an equation, and we explain the mathematical and biological conditions which made possible a good model fitting. Some other growth models in several detailes reached more complex results [10] but our aim is to develope a generally applicable model.

2. Accumulation types

Investigating of the accumulation process we have taken into account several points of view, including ideas of other authors, too.

One of the main factors resulting in differences is the nature of the nutrient element. We demonstrate the differences between the accumulation of dry matter, N, P and K using the material of paper [3] (see fig. 1). Five different curves correspond to five different treatments. Beyond the differences between the accumulation of different elements one can see that by varying the treatment rather different uptake and reflux processes can be obtained.

Another source of the variability of the accumulation process is the great variety of the crop species investigated. Figure 1 show some cases (dry matter, N, P) without reflux and processes with more or less intensive reflux (potassium) as well. This latter case is typical for the winter wheat. Paper [9] deals with the phenodynamics of the potato. Figure 2 refers to the results of this paper. It is specific for the potato (and for some other plants, e.g. pea) that a relatively long saturation period - plato - is followed by an abrupt reflux (collaps of the leaves). Figure 2 shows the effect of the fertilization, too. So the third 'dimension' of our model should be the effect of the different treatments.

The fourth 'dimension' is the natural environment. Paper [4] gives a comprehensive study of the habitats of the winter wheat in Hungary. Our example shown on figure 3 is taken from this work. It demonstrates the dependence of the nutrient uptake process on the agroecological environment, on the climatic effects, etc. The mean and extremal curves are drawn for different types of dynamics (those with reflux and without reflux).

This great variability and the low number of observations for a given soil-plant system (measurements were made at 4 or 5 phenophases) made inaccurate the model identification - moreower some other mathematical models (e.g. [10]) were inapplicable. Therefore we had to use some additional constraints in order to get correct estimates for the model parameters.

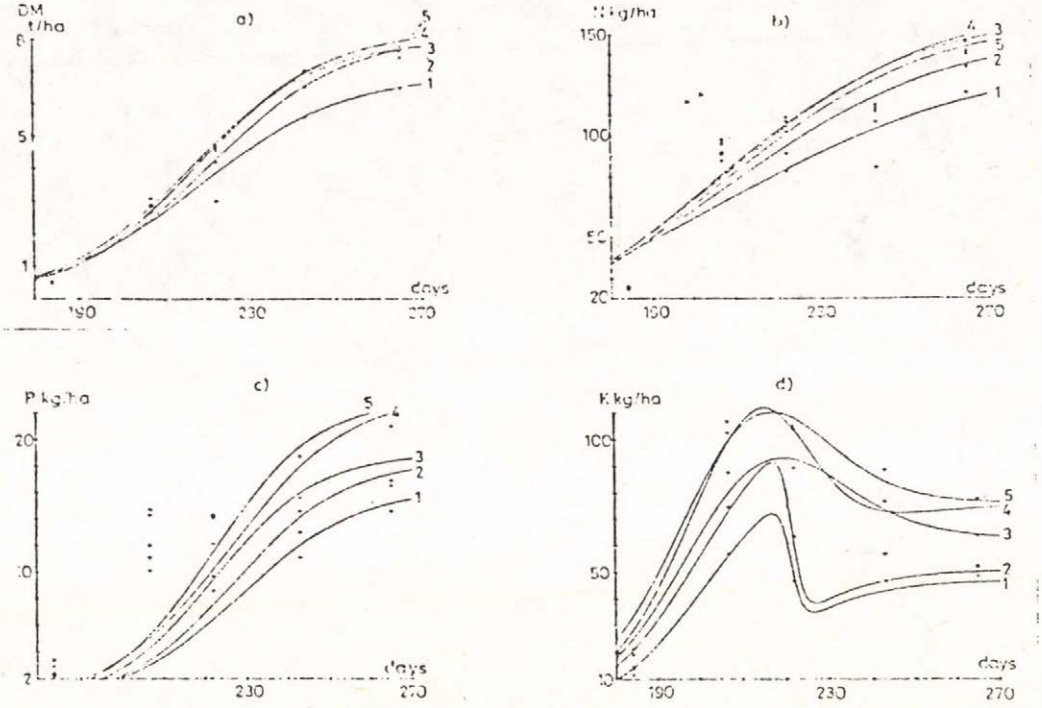


Figure 1. Growth and nutrient uptake by winter wheat

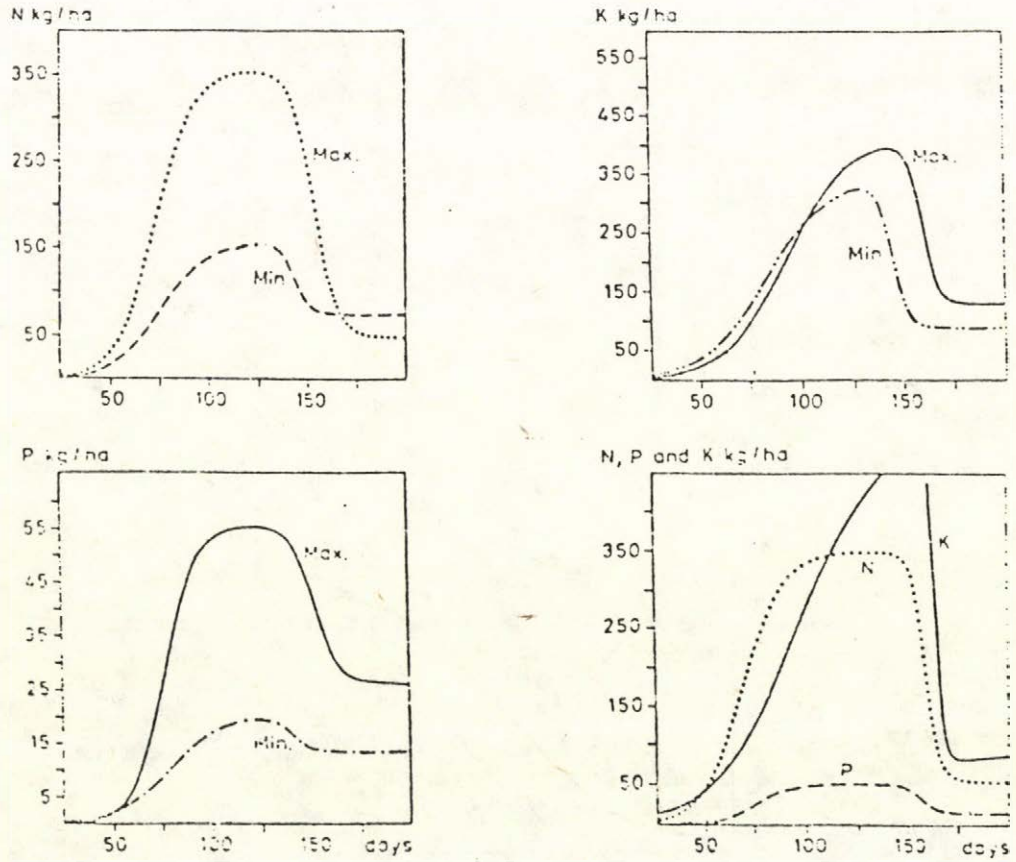


Figure 2. Mineral nutrient accumulation of potatoes

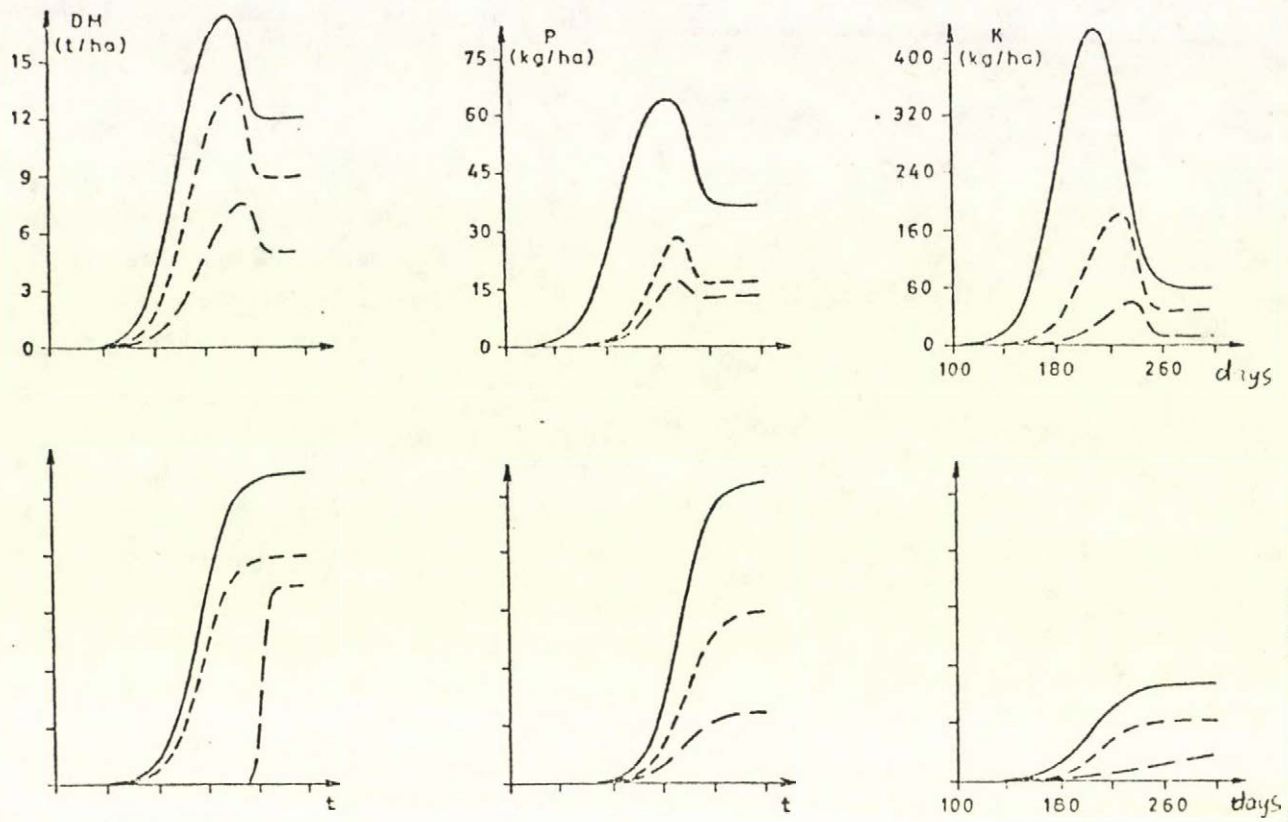


Figure 3. Range and actual mean of dry matter (DM) and P, K accumulation dynamics for winter wheat (types with reflux and without reflux)

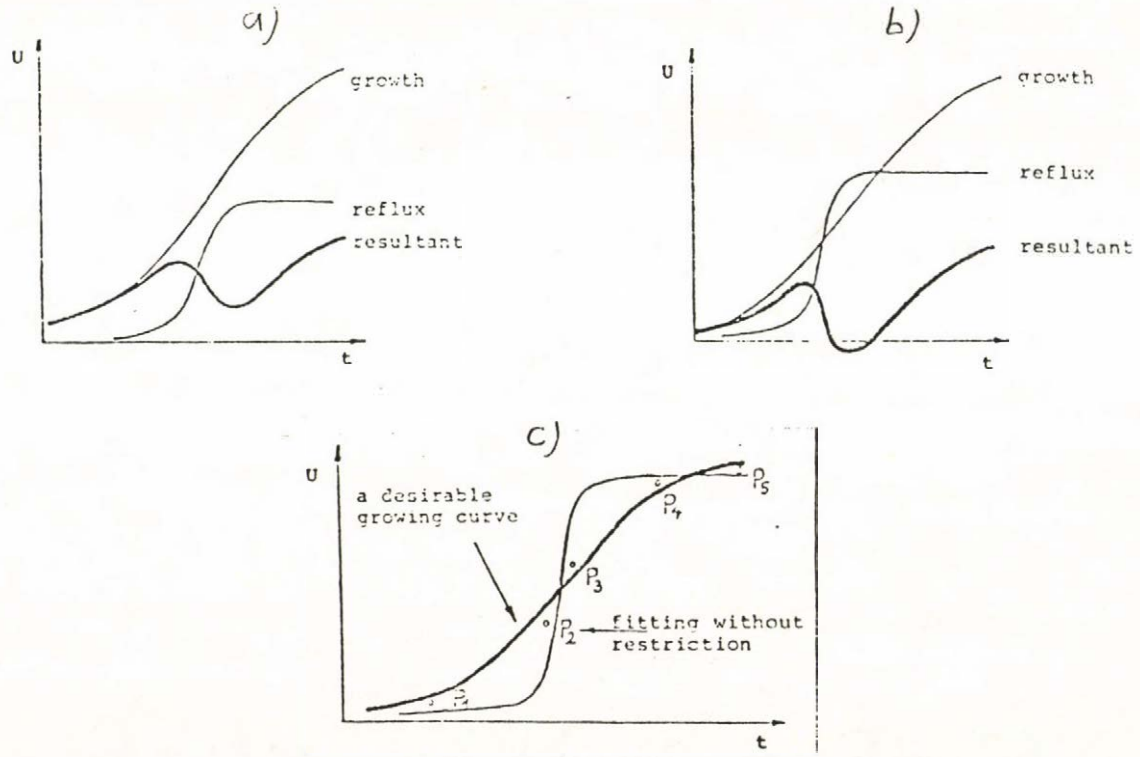


Figure 4. Fitting problems

3. The model fitting

During the model verification we encountered certain problems which will be discussed now.

In the majority of the cases analysed so far the number of points measured was only four or five, while there are six model parameters (A, R, b, t_g, s, t_s). In addition, it was necessary to consider the fact that certain natural restrictions are valid for the model:

(1) The model parameters are obviously positive.

(2) A>R , U>0.

(3) The derivative of U can only have one local minimum and maximum. This condition can, of course, be satisfied only approximately due to the fluctuations during growth or during nutrient element accumulation.

Total model fitting to four or five points was possible because the value A-R could easily be estimated from the data themselves. The analysis was always performed for fixed A values only. The R values can also be fixed if we know A-R. The lower limit for the real A value has derived from the measured maximum. The upper limit was controlled by genetic laws, according to Table 1, from which A< max value * (PMY/MAY). In this interval a sequence of values A was selected and a fitting procedure for the other parameters was performed for each fixed A value. The value corresponding to the best approximation (in the sense of least squares) was thereafter accepted.

Conditions (2) and (3) mentioned in connection with model parameters control the relationship between growth and the reflux (decomposition) process. The reflux cannot culminate before a saturation in the growth (counterexamples see on the fig. 4/a and fig. 1. - case potassium) and , in addition, the reflux cannot exceed the growth (U>0, counterexample in the fig. 4/b).

Sometimes, if the number of data is small, it may happen that the 'optimal' curve (in the sense of least square) have unreal b or s parameter values (fig. 4/c); they turn out to be too large from the point of view of plant physiology. Fortunately the maximum values of b and s can be estimated using the fact that the phenophases which determine the dominant time interval of the growth (accumulation) can be well fixed (table 2.).

Table 1. Potential production limits adopted in the model for the arable crops observed (t/ha)

Species	MAY	PMY	PMY/MAY
Winter wheat	4.28	10.0	2.33
Winter barley	3.38	7.5	2.22
Rye	1.75	6.0	3.43
Maize	5.42	14.0	2.58
Potato	16.20	50.0	3.09
Green pepper	11.52	21.0	1.82
Red pepper	6.00	11.0	1.83
Sugar beet	34.35	80.0	2.33
Peas	2.33	6.5	2.79
Lupin	25.00	60.0	2.40

MAY = maximum average yield measured in Hungary
PMY = practicable maximum yield

Table 2. Mean length of intensive growth periods in Hungary

Plant	Phenophase	Day of the vegetation season	Max. difference of phenophase
Winter wheat	Stalk formation	160-193	7 (days)
Winter barley	Stalk formation	140-170	4
Rye	Stalk formation	180-200	5
Maize	Tasseling	60-90	4
Potato	Beginning of flowering	60-90	4
Green pepper	Flowering	70-90	5
Red pepper	Flowering	70-100	5
Sugar beet	Root swelling	85-115	5
Pea	Flowering	60-90	7
Lupin	Flowering	70-100	5

References

1. Békéssy A., Biczók Gy., Ruda M., Modelling the dynamics of crop nutrient uptake, in: J. Badia et al. (eds.), Abstracts of Contributed Papers, XIth International Biometric Conference, Toulouse, 1982, p. 6.
2. Biczók Gy., Elek É., Békéssy A., Ruda M., Analysis of agroecological systems and modelling their nutrient cycles, in: Systems Science VIII; International Conference on Systems Science - Abstracts of Papers, Wroclaw, 1983, pp. 16-17.
3. Biczók Gy., Lásztity B., Relationship between nutrient uptake by winter wheat and nutrient supply of soil, I. Theoretical Approach, in: Zbornik Radova, Vol. 14, 1984, pp. 21-27.
4. Biczók Gy., Lásztity B., Békéssy A., Ruda M., Modelling of Dry Matter and Nutrient Accumulation by Winter Wheat, in: Proceedings of the Hugarian-British Joint Seminar, Session B, Soil Fertility, Agrochemistry and Soil Science, Vol. 34 Suplementum, 1985, pp. 108-119.
5. Biczók Gy., Lásztity B., Békéssy A., Krámli A., Ruda M., Discussion of a model for nutrient uptake of arable crops, in: M.H. Hamza (ed.), Modelling, Identification and Control, IASTED Publication, Acta Press, 1987, pp. 71-74.
6. Láng I., Csete L., Harnos Zs., The Agroecological Potential of the Hungarian Agricultura in the turn of the millennium (in Hungarian), Mezőgazdasági Kiadó, Budapest, 1983.
7. Lásztity B., Biczók Gy., Modelling on the kinetics of dry matter and nutrients accumulation in: winter barley, in: Agrochimica, Vol. XXVII, No. 5-6, 1983, pp. 514-520.
8. Lásztity B., Biczók Gy., Ruda M., Evaluation of Dry Matter and Nutrient Accumulation in: Winter Wheat, in: Cereal Research Communications, Vol. 12, No. 3-4, 1984, pp. 193-199.
9. Németh T., Biczók Gy., Ruda M., The effect of nitrogen fertilization on the yields and mineral nutrient accumulation of potatoes, in: E. Welte, I. Szabolcs (eds.), Fight Against the Hunger throught Improved Plant Nutrition, 9th World Fertilizer Congress Proceesdings, CIEC, Belgrade, Goettingen, Vienna, 1984, Vol. 3, pp. 195-196.
10. Zelawski W., Lech A., Logistic growth functions and their applicability for characterizing dry matter accumulation in plants, in: Acta Physiologia Plantarum, Vol. 2, 1980, pp. 187-194.

A TANULMÁNYOK SOROZATBAN 1987-BEN MEGJELENTEK:

195/1987 Telegdi László: Bináris változók strukturájának vizsgálata

196/1987 Rónyai Lajos: Algebrai algoritmusok

197/1987 Hernádi Ágnes - Bodó Zoldán - Knuth Előd: A tudásábrázolás technikái és gépi eszközei

198/1987 Miguel Fonfria Atan: A data base management system developed for the Cuban minicomputer CID 300/10

199/1987 Bach Iván - Farkas Ernő - Naszódi Mátyás: A magyar nyelv elemzése számitógéppel

200/1987 PUBLIKÁCIÓK - Publications 1986
Szerkesztette: Petróczy Judit

201/1987 Eszenszki József - Hévizi László - Kas Iván - Laufer Judit - Palotási András - Szőnyi Tamás - Vörös Károly: Tanulmányok a számitástechnika nyomdaipari alkalmazásához

202/1987 PROBLEMS OF COMPUTER SCIENCE
Proceedings of the joint workshop of Computer and Automation Institute of HAS and Computing Centre of Armenian Academy of Sciences held in Budapest, September, 1987.
Edited by G.B. Marandzjan, B. Uhrin

Készült a Technografik gondozásában
Az OMIKK nyomdájában
1011 Budapest, Gyorskocsi u. 5-7.
Felelős vezető: Tóth Károly